Nunes
Liliana Vechina O problema de planeamento de redes com custo ótimo para o utilizador

Universidade de Aveiro Departamento de Matemática
2012

Liliana Vechina Nunes

O problema de planeamento de redes com custo ótimo para o utilizador

Universidade de Aveiro Departamento de Matemática
2012

Liliana Vechina Nunes

O problema de planeamento de redes com custo ótimo para o utilizador

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Matemática Aplicada, realizada sob a orientação científica da Doutora Rosa Maria Videira de Figueiredo, Investigadora auxiliar da Universidade de Aveiro.

o juri / the jury

presidente / president

Doutor Agostinho Miguel Mendes Agra
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Doutora Maria Adelaide da Cruz Cerveira
Professor Auxiliar da Universidade de Trás-os-Montes e Alto Douro

Doutora Rosa Maria Videira De Figueiredo (Orientadora)
Investigador Auxiliar da Universidade de Aveiro

agradecimentos / acknowledgements

Em primeiro lugar, à Dra. Rosa Maria Videira De Figueiredo, com quem tive o prazer de trabalhar nesta dissertação de mestrado. Agradeço a sua dedicação e disponibilidade e acima de tudo, as suas palavras de encorajamento e estímulo. À universidade de Aveiro, e em particular ao Departamento de Matemática pela oportunidade e pelas condições proporcionadas para desenvolver este trabalho. À minha família e amigos, em especial ao Bruno Queirós e ao António Neves pelo apoio, compreensão, paciência e pelo estímulo constante à conclusão deste trabalho. A todos os que, direta ou indiretamente, contribuíram para este trabalho.

Palavras-chave

Planeamento de redes; Caminho mais curto; Programação binária em dois níveis; Algoritmo Enumerativo.

Resumo

Nesta dissertação estuda-se o planeamento de redes com custo ótimo para o utilizador. Uma das aplicações deste problema é o transporte de materiais perigosos, que devido aos seus elevados níveis de risco, sensibilizam a população para esta atividade. Existem alguns métodos exatos e heurísticos para resolver o problema. Um método exato consiste em adaptar um algoritmo enumerativo proposto na literatura para resolver um problema de programação linear inteira em dois níveis. Nesta adaptação problemas de caminho mais curto surgem como subproblemas que necessitam ser resolvidos de forma eficiente.

Keywords Network planning, Shortest path, Discrete bilevel programming problem, Branch-and-Bound.

Abstract

In this work we study a network planning problem with optimal cost for the user. One application of this problem is the transportation of hazardous materials. In the literature, we find some exact and heuristics methods to solve this problem. In this work another exact method to this problem is studied: we adapt an enumerative algorithm proposed in the literature to the solution of 0-1 bilevel linear programming problems. In this adaptation shortest path problems arise as subproblems that need to be solved efficiently.

– Liliana Vechina Nunes

Conteúdo

Lista de Figuras

Capítulo 1

Introdução

A Otimização é uma área que estuda a procura pela melhor solução possível para um determinado problema. Um problema de otimização consiste, de grosso modo, em minimizar ou maximizar uma função, de forma a encontrar a solução ótima num determinado domínio, domínio esse normalmente definido por um conjunto de restrições. A Combinatória estuda arranjos, agrupamentos, ordenações ou permutas de elementos discretos, que em geral constituem um conjunto finito dotado de estruturas particulares. A Otimização Combinatória é um ramo da Ciência da Computação e da Matemática Discreta que se baseia no uso conjunto de técnicas de combinatória, programação matemática e teoria de desenvolvimento de algoritmos, para resolver problemas de otimização formulados sobre estruturas binárias.

Problemas de Otimização Combinatória ocorrem em áreas tão diversas como:

- Projetos de sistemas de distribuição de energia elétrica,
- Posicionamento de satélites,
- Rotas ou escalonamento de veículos,
- Alocação de trabalhadores ou máquinas a tarefas,
- Corte e empacotamento de caixas em contentores,
- Sequências de genes e DNA, entre outros.

Existem vários métodos para resolução deste tipo de problemas, como os métodos de Programação Linear e Não Linear, PLI, Programação Dinâmica e diversos métodos Heurísticos.

Nas sociedades modernas, procurar e fazer conexões é importante em muitos tipos de rede, como redes de telecomunicações ou transporte. O desenvolvimento urbano, obriga ao transporte entre diversos locais de uma quantidade cada vez maior de pessoas e produtos, como matéria-prima para a indústria, material perigoso (como gasolina, produtos radioativos e lixo) que deve ser conduzido de forma segura pelos centros urbanos. Cada vez mais se exige que este transporte seja feito de forma rápida e barata. Os problemas de planeamento de redes surgem para ajudar a encontrar formas eficientes de se realizarem esses transportes, no fundo para ajudar no desafio de transportar cada vez mais e melhor. Estes atraem a atenção pela sua aplicação sobretudo na logística e na área de telecomunicações.

Problemas de caminho mais curto (CMC) fazem muitas vezes parte de subrotinas dentro de procedimentos para resolver problemas de planeamento de redes. Os caminhos são dos mais básicos e importantes objetos de estudo da Otimização Combinatória. Estão presentes no uso prático diário, para fazer conexões e procuras. Podemos mesmo ver que nas sociedades mais primitivas, encontrar caminhos curtos e fazer procuras (por comida, por exemplo) era essencial.

Curto não significa apenas em termos de distância geométrica, mas pode ter em conta outros fatores.
Quando há restrições ao caminho a escolher, ocorre outra variante de problemas de CMC que se denomina por problemas de CMC com Restrições Adicionais (PCMCRA). Estas restrições podem ser de diversos tipos, como por exemplo:

- Restrições de Recurso. Neste caso o problema consiste em encontrar um caminho de menor custo, satisfazendo as capacidades máximas disponíveis dos vários recursos. Assume-se que percorrer um arco implica consumir um recurso escasso, por isso, não se pode ultrapassar a capacidade máxima desse recurso.

- Restrições que impõem janelas temporais nos vértices. Neste caso, o problema consiste em encontrar um caminho de menor custo entre dois vértices de um grafo, em que aos arcos estão associados tempos de percurso e a cada vértice existe uma janela temporal associada, que garante que o vértice só é visitado num dado intervalo de tempo.

Este é o tema desta tese, o problema de planeamento de redes com custo ótimo para o utilizador.

O problema estudado em [11], consiste no planeamento de redes onde o utilizador faz o transporte pelo CMC. Vamos denominar esse problema de *Fixed Charge Network Design Problem with Shortest Path Constraints* (FCNDP-SPR). O FCNDP-SPR envolve dois tomadores de decisão distintos agindo não cooperativamente e de uma forma sequencial. Por isso, é formulado como um problema de programação misto discreto linear em dois níveis. Apesar do *Fixed Charge Network Design Problem* (FCNDP) ter sido amplamente estudado, o FCNDP-SPR apenas foi tratado poucas vezes na literatura como em [14, 5]. A inclusão de restrições de CMC num programa inteiro misto traz dificuldades tanto na modelação, como na conceção de métodos de solução eficientes.
Substituindo o problema de nível inferior pelas suas condições de otimalidade, obtém-se uma formulação inteira em um nível para o problema FCNDP-SPR. Usando as condições de otimalidade de *Bellman* para o problema de CMC, obtém-se uma outra formulação inteira de um nível para este problema. O problema estudado nesta tese é exatamente o problema estudado em [11]. Contudo, em [11] os autores resolveram este problema usando uma heurística de busca Tabu. Neste trabalho vamos descrever um algoritmo de enumeração implícita para resolver o FCNDP-SPR.

1.1 Motivação para o problema

O problema que vamos estudar possui aplicação direta no planeamento de redes de transporte público, no transporte de produtos de risco e aplicação indireta na área de telecomunicações. A imposição no projeto de redes de condições de otimalidade para o utilizador, torna mais difícil tanto o processo de modelação como o desenvolvimento de métodos de solução para o problema.

Um exemplo habitual deste problema é o transporte de materiais perigosos. Durante a última década, o transporte de materiais perigosos atingiu níveis bastante elevados. Muitos dos materiais transportados por camiões, comboios, embarcações e aviões são inflamáveis, explosivos, tóxicos, corrosivos ou radioativos. Devido à natureza da carga, estes transportes estão associados a significativos níveis de risco. Apesar de serem potencialmente nocivos para o ambiente e para as pessoas, são essenciais para o desenvolvimento industrial, pelo que continua a existir um elevado potencial de incidentes catastróficos sociais e ambientais associados a essas transferências, que têm uma grande magnitude de consequências indesejáveis, como múltiplas mortes, lesões, grandes evacuações e graves danos ambientais. Por isso, a regulamentação de expedições de mercadorias perigosas é geralmente da responsabilidade do governo. Para além de outras formas de mitigação deste problema, é feito o treino de condutores, estabelecem-se estações de

inspeção para monitorar a conformidade dos regulamentos e fornecem-se sistemas de resposta de emergência para minimizar as consequências dos incidentes. O governo também tem autoridade para impedir o transporte de certos materiais perigosos em algumas estradas da rede. No fundo, pretende-se limitar este transporte a um subconjunto de estradas disponíveis. Assim, é o Estado que tem o poder de escolher a rede de estradas que os transportadores poderão posteriormente usar.

As empresas transportadoras e os governos têm perspetivas muito diferentes em relação ao transporte de mercadorias perigosas. As transportadoras procuram naturalmente minimizar os custos das rotas utilizadas, enquanto os governos têm como objetivo a redução dos riscos (do público e do meio ambiente), sem ameaçar a viabilidade da economia com o transporte de materiais perigosos. Para além disso, o governo tem que considerar uma variedade de produtos perigosos e um grande número de pares de origem-destino (OD). Um transportador, no entanto, pode planear cada embarque separadamente. Uma esmagadora maioria do transporte de materiais perigosos na literatura adota o ponto de vista da transportadora, focando problemas com um único material perigoso e um único par OD. O tema principal dos estudos existentes é a avaliação do risco associado ao transporte de uma remessa e a procura pela rota que minimiza este risco.

O problema que considera a minimização do risco por parte do governo e simultaneamente a minimização dos custos de transporte por parte das transportadoras, começou a ser estudado na literatura académica por *Bahar Y. Kara* e *Vedat Verter* [14]. Este foi o primeiro trabalho a colocar o problema de projeto de redes de transporte de materiais perigosos como um problema de PLI em dois níveis. Analisa-se o problema de selecionar uma rede de estradas para o transporte de mercadorias perigosas a partir de uma infraestrutura de transportes existente. Eles assumem que as transportadoras, representadas pelo seguidor (segundo nível) no modelo de dois níveis, usam sempre as rotas mais baratas na rede de transportes de materiais perigosos projetada pela autoridade governamental, que assume o papel de líder (primeiro nível), e tem o objetivo de selecionar o risco mínimo da rede total. Estes autores adotaram o ponto de vista do governo.

1.2 Revisão da Literatura

Tudo que foi feito para o problema de planeamento de redes com custos ótimos para o utilizador, foi no contexto de transporte de materiais perigosos. Então será apresentada de seguida uma revisão dessa literatura.

Como dito na subseção anterior, em [14] o problema é formulado como um problema inteiro misto de dois níveis onde os modelos problemáticos do líder são as decisões do governo, enquanto que o problema interno corresponde ao percurso selecionado pelo transportador.

No trabalho [2], prova-se que a versão do problema de transporte de materiais perigosos em redes onde um subconjunto de arcos pode ser proibido é NP-difícil, mesmo quando apenas é enviada uma única mercadoria. Propõe-se uma formulação de PLI em dois níveis e deriva-se uma formulação de PLI mista em um nível que pode ser resolvida em tempo razoável, e é mais compacta do que em [14].

Erkut e Alp [9] consideram um problema em um único nível. Eles limitam a rede a uma árvore, de modo que exista um único caminho entre cada origem e destino. Assim, não têm caminhos alternativos sobre a árvore. Eles formulam o problema de projeto da árvore como um problema de PLI, com o objetivo de minimizar o risco total de transportes. O seu modelo pode ser resolvido para problemas de tamanho moderado. Desenvolvem uma heurística gananciosa de construção simples, para expandir a solução do problema de projeto da árvore, adicionando segmentos da estrada de forma incremental, o que permite às autoridades locais a troca entre

risco e custo.

Erkut e *Gzara* [10] em 2008 consideram um problema semelhante ao de *Kara* e *Verter* [14], generalizando o seu modelo, onde consideram uma rede não orientada. Os investigadores transformam o programa de dois níveis num problema linear inteiro misto de apenas um nível, substituindo o problema de segundo nível pelas suas condições KKT e pela linearização das restrições de complementaridade de folga. Propõem um método de solução heurístico. Além disso, estendem o modelo de dois níveis para ter em conta o custo/risco de *trade-off*, pela inclusão de custo na função objetivo do problema do líder (primeiro nível). A redução de risco vem com um significativo aumento no custo. Uma vez que o governo designa a rede, as transportadoras terão as rotas de menor custo entre a origem e o destino nesta rede.

Em [4], é fornecida uma formulação de programação linear em dois níveis, que tem em conta tanto a minimização de risco total como a equidade do risco. Transformam o modelo de dois níveis num programa linear inteiro misto de um único nível, tal como *Kara* e *Verter*. Uma vez que o modelo de dois níveis também é difícil de resolver de forma otimizada, fornecem um algoritmo heurístico para este modelo.

No trabalho de Keeney [15], o problema tratado é ligeiramente diferente. Formula-se o problema como um modelo linear de dois níveis, onde no nível mais elevado (líder), há uma meta-autoridade local agindo em nome de todas as autoridades locais envolvidas, isto é, a igualdade de risco, e no nível mais baixo (seguidor), há a autoridade da área regional que visa minimizar o risco total sobre a rede. Isto corresponde à existência de dois decisores: a autoridade regional e as autoridades locais.

Na maior parte da literatura, o problema é orientado ao ponto de vista das transportadoras. Conhecemos duas notáveis exceções. O modelo de *Lista* e *Mirchandani* [16], para o roteamento de materiais perigosos, minimiza o custo total, risco social total e risco máximo imposto sobre um indivíduo. *Iakovou* em [13], fornece um modelo de fluxo de rede multimercadoria cujo objetivo é evitar sobrecarregar determinadas ligações da rede de transporte.

Nesta dissertação vamos resolver o FCNDP-SPR, estudado em [11], utilizando um algoritmo de enumeração implícita proposto em [17]. Para resolver os problemas de CMC que surgem como subproblemas no FCNDP-SPR, vamos usar algoritmos conhecidos na literatura para problemas de CMC como o *Dijkstra* [1] e um algoritmo de programação dinâmica apresentado em [18] para problemas de CMC com restrições adicionais.

O trabalho está organizado em 5 grandes capítulos: conceitos básicos, definição do problema, algoritmo exato para o problema de programação linear binário em dois níveis, algoritmos para resolução de problemas de CMC e o algoritmo para o nosso problema. No capítulo 2, é feita uma breve referência a conceitos necessários ao longo da tese. No capítulo 3, é apresentada a definição do nosso problema em dois modelos: o modelo linear binário misto em dois níveis e a passagem deste ao modelo inteiro linear a um nível. De seguida, no capítulo 4, é feita a descrição de um algoritmo exato para o problema de programação linear binária em dois níveis. No capítulo 5, descreve-se o problema de CMC, algumas das suas variantes que incluem restrições adicionais e é mostrada a abordagem da Programação Dinâmica para um desses problemas. Finalmente, no capítulo 6, é apresentado o algoritmo exato para o FCNDP-SPR, como adaptação do algoritmo do capítulo 4, que irá usar algoritmos apresentados no capítulo 5 para resolver os sub-problemas.

Segue uma lista de algumas siglas usadas ao longo do texto:

- BLPP - *Bilevel Linear Programming Problem.*
- CMC - Caminho mais curto.
- CMCRA - Caminho mais curto com restrições adicionais.
- CMCRC - Caminho mais curto com restrições de capacidade.
- FCNDP - *Fixed Charge Network Design Problem.*
- FCNDP-SPR - *Fixed Charge Network Design Problem with Shortest Path Constraints.*
- PLI - Programação linear inteira

Capítulo 2

Conceitos Básicos

Neste capítulo iremos dar os conceitos básicas de teoria de grafos, problemas lineares e outras definições relevantes, bem como apresentar alguma notação que será usada ao longo do trabalho.

2.1 Grafos

Nesta seção introduzimos algumas noções elementares da teoria de grafos.

Um grafo não orientado G é um par $G = (V, E)$ onde $V = \{1, 2, ..., n\}$ é um conjunto finito e não vazio de vértices e E é um conjunto também finito de pares distintos e não ordenados de elementos de V, $E = \{\{i, j\} : i, j \in V, i \neq j\}$. Os elementos de V designam-se nós, nodos ou vértices. Os elementos de E designam-se arestas.

Graficamente, um grafo pode ser representado como na figura seguinte, onde os vértices são representados por círculos e as arestas por um segmento de reta.

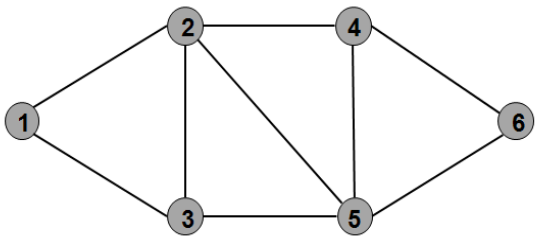

Figura 2.1: Grafo não orientado.

Graficamente, no caso dos grafos serem orientados, os arcos são representados por setas. Um grafo orientado, ou **digrafo**, $G = (V, A)$ consiste num conjunto V de vértices e um conjunto A de arcos, que são pares ordenados de elementos distintos de V. Ou seja, aos elementos (i, j) de A está associada uma direção de i para j.

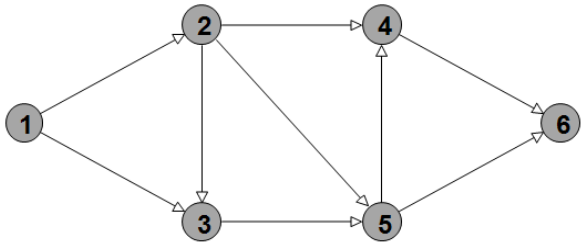

Figura 2.2: Grafo orientado ou digrafo.

A maior parte das definições será apresentada apenas para grafos orientados. Mas todas elas podem facilmente ser adaptadas para o caso de grafos não orientados. Para mais detalhes sobre as definições o leitor deverá ver [1].

Uma aresta $\{i, j\}$ ou arco (i, j) diz-se **incidente** no vértice i e no vértice j. Neste caso, dizemos que os vértices i e j são **adjacentes**. A **Lista de arcos de incidência** é dada por $B(i) = \{(i, j) \in A : i, j \in V\}$.

Sejam s e t dois vértices de um grafo $G = (V, A)$, respetivamente vértice inicial e vértice final. Neste texto, um **passeio** de s para t num grafo orientado $G = (V, A)$ é um subgrafo de G caracterizado por uma sequência constituída alternadamente por vértices e arcos da forma $p = v_1 a_1, \ldots, a_{k-1} v_k$ com $k \geq 2$, onde:
$v_1, \ldots, v_k \in V$;
$v_1 = s$ e $v_k = t$;
$a_1 \ldots a_{k-1} \in A$ onde $a_i = (v_i, v_i + 1) \in A$, $i = \{1, \ldots, k-1\}$. Um **caminho** de s para t em

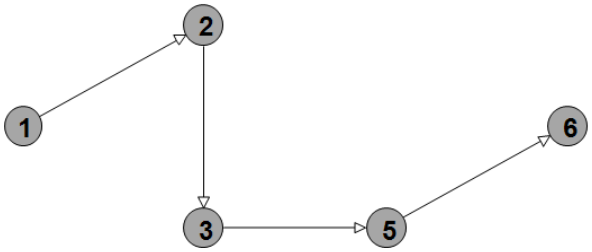

Figura 2.3: Passeio.

(V, A) é um passeio com início em s e fim em t, onde:
$v_i \neq v_j$, para $i, j \in \{2, ..., k-1\}$ tais que $i \neq j$;
$v_i \neq s$ e $v_i \neq t$, para $i \in \{2, ..., k-1\}$.
Isto é, um caminho é um passeio sem qualquer repetição de vértices intermédios (na Figura 2.3 o passeio representado é também um caminho).

Um **ciclo** é um caminho fechado, isto é, um caminho de um qualquer vértice para ele mesmo.

Como dito anteriormente, os conceitos de caminho e ciclo são facilmente adaptados para grafos não orientados, substituindo arcos por arestas nas definições apresentadas.
Um **grafo não orientado** diz-se **conexo** se entre cada par de vértices $i, j \in V$ existe um caminho que une i a j. Caso contrário, o grafo diz-se desconexo.

Na Figura 2.5, apresenta-se um exemplo de um grafo desconexo.

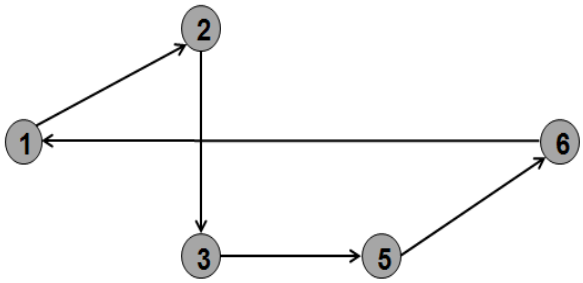

Figura 2.4: Um ciclo.

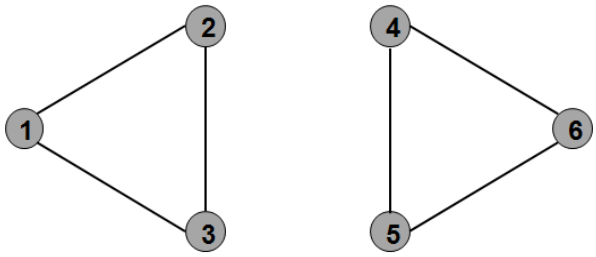

Figura 2.5: Grafo desconexo.

Um digrafo $G = (V, A)$ diz-se **conexo** se o grafo $G' = (V, A')$ obtido a partir de G removendo a orientação dos arcos de A for conexo.

Uma **árvore** é um grafo conexo e sem ciclos.

Vamos assumir agora que a cada arco (i, j) do grafo orientado $G = (V, A)$ está associado o valor c_{ij} que é o comprimento do arco (i, j). Considere-se o caminho p de s para t no grafo G. O comprimento do caminho p, $c(p)$, é a soma dos comprimentos dos arcos que pertencem àquele caminho, isto é, $c(p) = \sum_{(i,j)\in P} c_{ij}$.
O conjunto de todos os caminhos de s para t identifica-se por P. Ao caminho de menor comprimento em P dá-se o nome de **Caminho Mais Curto**.

2.2 Problemas Lineares

Nesta seção é feita uma revisão de alguns conceitos básicos como Programação Linear, de Programação Linear Inteira e programação linear binária a dois níveis. Esta seção é baseada em [19].

Os problemas de **Programação Linear** (PL) são problemas de otimização nos quais a função objetivo e as restrições são descritas por funções lineares. Geometricamente, as restrições lineares definem um poliedro, que é chamado de conjunto das soluções admissíveis. Assim, um poliedro P é um subconjunto de $\mathbb{R}^n$ descrito por um número finito de desigualdades lineares: $P = \{x \in \mathbb{R}^n : Ax \leq b\}$.

Estes problemas podem ser definidos da seguinte forma:

$$\text{minimizar/maximizar } cx \tag{2.1}$$

Sujeito a

$$Ax \leq b, \tag{2.2}$$

$$x \geq 0. \tag{2.3}$$

onde $c \in \mathbb{R}^{1\times n}$, $A \in \mathbb{R}^{m\times n}$ e $b \in \mathbb{R}^{m\times 1}$ são dados de entrada do problema e $x \in \mathbb{R}^{n\times 1}$ representa a sua variável de decisão. A função linear a ser minimizada/maximizada em (2.1) é denominada função objetivo. As restrições de não negatividade (2.3) são conhecidas como triviais.

Quando nos problemas de PLI obrigamos todas as variáveis de decisão a só admitirem valores inteiros, estaremos diante de um problema de **Programação Linear Inteira** (PLI). São portanto problemas de programação matemática, em que a função objetivo, bem como as restrições, são lineares, porém todas as variáveis de decisão devem apenas assumir valores inteiros. Quando apenas uma parte das variáveis são do tipo inteiro, enquanto outras são do tipo real, dizemos que se trata de um problema de **Programação Linear Inteira Mista**. Quando as variáveis estão limitadas a tomar apenas os valores 0 ou 1, então temos um problema de **Programação Linear Binária**.

Poderia parecer que os problemas de PLI são relativamente fáceis de resolver, já que existem muito menos soluções a serem consideradas do que num problema de PLI. A simples introdução das restrições de integralidade das variáveis num problema de PL, transforma-o num problema de características diferentes. Em geral, os problemas de PLI são NP-difíceis.

Outro tipo de problema linear é o problema de **programação linear em dois níveis**, mais conhecido por *Bilevel Linear Programming Problem* (BLPP). É um exemplo de um jogo não cooperativo de duas fases, no qual o primeiro jogador pode influenciar mas não controlar as ações do segundo. No fundo, é uma otimização simultânea de dois objetivos. O 0-1 BLPP é um modelo para o jogo líder-seguidor (ou comummente também denominado jogo de *Stackelberg*), em que dois jogadores tentam maximizar as suas funções objetivo individuais, $F(y,x)$ e $f(y,x)$, respetivamente. As variáveis de decisão, x e y, são divididas entre os jogadores de forma que nenhum pode dominar o outro na sua escolha. O líder joga primeiro e é após a sua escolha de $y \in R^{n_1}$, que está apto para influenciar, mas não controlar, as ações do seguidor. De seguida, é a vez de jogar o seguidor. Ele escolhe um $x \in R^{n_2}$, já com o seu conjunto de escolhas possíveis reduzido anteriormente pelo líder, e com o objetivo de maximizar o seu lucro. Desta maneira, ele afeta indiretamente a escolha do líder.

Vamos considerar a versão linear binária deste jogo de *Stackelberg* denotada por 0-1 BLPP. Qualquer solução deste problema tem de pertencer ao conjunto que chamaremos de conjunto de reação racional do seguidor, ou seja, deve fazer parte do conjunto de escolhas possíveis para o seguidor, dada uma escolha do líder. O problema é formulado como se segue:

$$\text{minimizar/maximizar}_y F(y,x) = c^1 y + c^2 x, \tag{2.4}$$

$$y \in Y = \{y_j \text{ binário}; j = 1, \ldots, n_1\}$$

onde x resolve

$$\text{minimizar/maximizar}_x f(y,x) = d^1 y + d^2 x, \tag{2.5}$$

Sujeito a

$$g(y,x) = Ay + Bx \leq b \tag{2.6}$$

$$x \in X = \{x_j \text{ binário}; j = 1, \ldots, n_2\} \tag{2.7}$$

onde c^1 e $d^1 \in \mathbb{R}^{1\times n_1}$, c^2 e $d^2 \in \mathbb{R}^{1\times n_2}$, $A \in \mathbb{R}^{m\times n_1}$, $B \in \mathbb{R}^{m\times n_2}$, e $b \in \mathbb{R}^{m\times 1}$. Os elementos destes vetores e matrizes são assumidos como constantes inteiras. Em (2.4) temos a função objetivo do

líder e em (2.5) a do seguidor. O problema do seguidor fica definido pelas restrições de (2.5) a (2.7). O termo d^1y, na função objetivo do seguidor, pode ser omitido sem afetar a solução do 0-1 BLPP, pois assim que o y é escolhido, o problema resultante [(2.5)-(2.7)] torna-se um programa linear binário apenas em x. Qualquer que seja a escolha do seguidor, d_1y não sofre alterações. Esta observação explica a diferença entre (2.5) e (4.11).

2.3 Outras definições

Uma **Heurística** é uma técnica para obter boas soluções, com um custo computacional aceitável, muitas vezes baseada em procedimentos simples e intuitivos. Não oferece garantias de otimalidade e pode nem sequer dar informação da qualidade de uma dada solução admissível. Baseia-se no compromisso entre a qualidade da solução e o tempo necessário para a obter.

Uma solução diz-se ótima no sentido de Pareto ou **Pareto-ótima** se não for possível melhorá-la, ou, mais genericamente, se não for possível melhorar a utilidade de um agente (no nosso caso uma função objetivo) sem degradar a situação ou utilidade de qualquer outro agente.

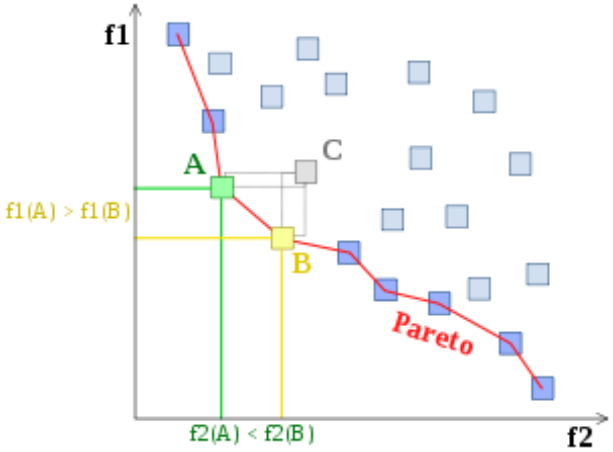

Figura 2.6: Fronteira de Pareto.

No gráfico anterior, pretendemos minimizar a solução. Dizemos que a solução A domina uma solução C, se as seguintes condições forem satisfeitas:

(i) A é pelo menos igual a C, em todos os objetivos, ou seja, $f_m(A) \leq f_m(C)$ para todo $m = 1, \ldots, M$ (a solução A não é pior que a solução C em nenhum dos objetivos).

(ii) A é superior a C em pelo menos um objetivo, ou seja, $f_m(A) < f_m(C)$ para algum m pertencente a $\{1, \ldots, M\}$ (a solução A é estritamente melhor que a solução C em pelo menos um objetivo).

Capítulo 3

O FCNDP-SPR

De seguida, iremos apresentar e formular o problema de planeamento de redes com custos ótimos para o utilizador como um 0-1 BLPP.

Seja $G = (V, E)$ um grafo não orientado, onde V é o conjunto de vértices desse grafo e E o conjunto de arestas admissíveis que os conectam. Para cada aresta $e \in E$ associamos um custo fixo de abertura da aresta f_e. Seja $A^E = \{(i,j),(j,i)|\ \forall\{i,j\} \in E\}$ o conjunto de arcos obtidos bidireccionando as arestas em E. Considere um arco $a = (i,j) \in A^E$ onde definimos $e(a) = \{i,j\}$ e $\overline{a} = (j,i)$. Para cada arco $a \in A^E$ associamos um custo positivo que representa o seu comprimento c_a e um custo operacional variável g_a^k para o transporte de um dado produto através dele (isto é, o custo variável de transportar o produto k pelo arco a), para cada produto $k \in K$. No nosso caso vamos considerar o caso simétrico em que os arcos (i,j) e (j,i) têm o mesmo comprimento c_a. Também considerando uma aresta $e = \{i,j\} \in E$ definimos $a(e) = (i,j)$ e $\overline{a}(e) = (j,i)$. Repare que não há capacidades associadas nem às arestas no conjunto E nem aos arcos no conjunto A^E. Considere ainda o conjunto K dos produtos a serem transportados nesta rede. Para cada $k \in K$, associamos um par de vértices (o^k, d^k) que representam, respetivamente, a origem e o destino em E do produto k.

De seguida será apresentado um exemplo para simplificar a compreensão da definição do problema. Para facilitar a compreensão do exemplo, todos os arcos possuem o mesmo comprimento, igual a 1. Seja $|K| = 2$. Desta forma, temos dois pares OD, respetivamente (1,4) e (2,5).

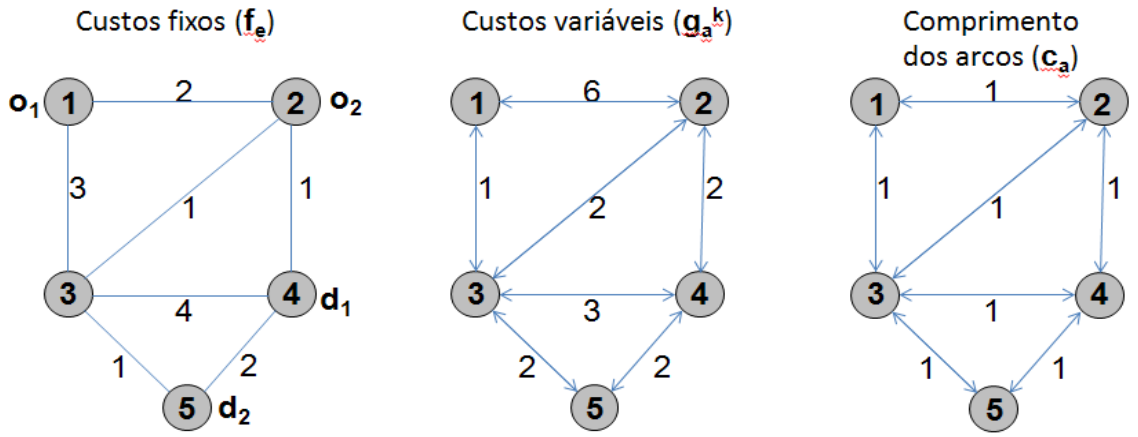

Figura 3.1: Rede.

Suponha-se que se decide, por algum método fechar as arestas (2,3) e (3,4), gerando a rede projetada na Figura 3.2.

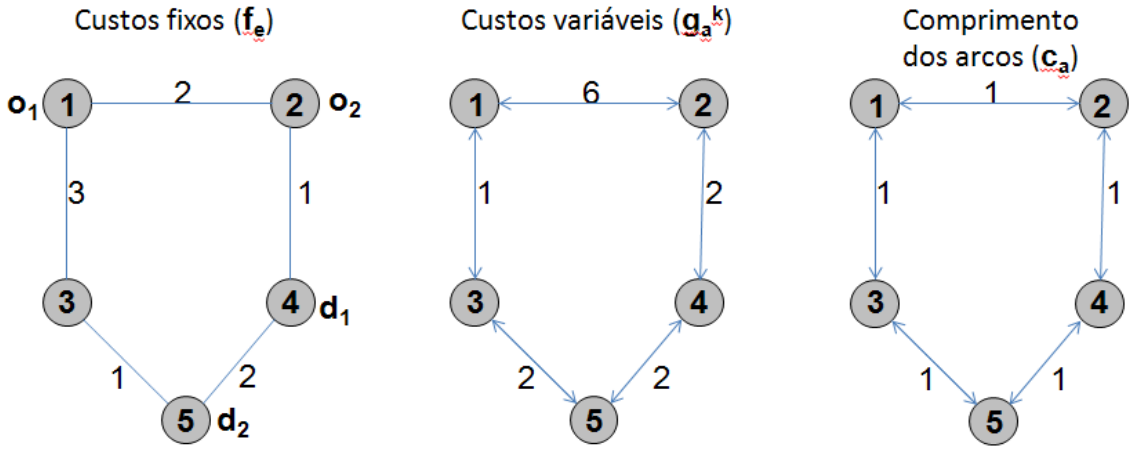

Figura 3.2: Rede projetada para o FCNDP.

O *Fixed Charge Network Design Problem* (FCNDP), amplamente estudado na literatura, consiste em selecionar um subconjunto de arestas de E para serem abertas de modo que seja possível transportar cada produto $k \in K$ das suas origens até aos seus destinos. O problema pretende minimizar o somatório dos custos fixos f_e (relacionados com a escolha das arestas) mais os custos variáveis g_a^k (referentes ao transporte dos produtos pelas arestas anteriormente selecionadas). No fundo, consiste em considerar apenas um conjunto de caminhos aceitáveis (economicamente admissíveis) para cada transportador. No caso do exemplo da Figura 3.2, a solução admissível do FCNDP será o caminho que minimizar a função objetivo, isto é, o que minimizar a soma dos custos fixos mais os variáveis. Logo, para o par $o_1 - d_1$ será $P_1 : 1 - 3 - 5 - 4$ e para o par $o_2 - d_2$ seria $P_2 : 2 - 4 - 5$. Para calcular o custo destas soluções, temos de calcular o custo de abrir as arestas utilizadas, que neste caso é 9; transportar a mercadoria 1 custa 5 de acordo com o caminho referido atrás e para a mercadoria 2 custa 4. Logo, o custo total desta solução admissível é 9+5+4=18 (soma dos custos fixos mais os custos variáveis).
Tanto os custos fixos como os variáveis são lineares e não são associadas capacidades às arestas. O FCNDP engloba uma grande quantidade de problemas de planeamento de redes. Pode ser modelado como um problema de PLI pertencente à classe dos problemas NP-Difíceis.

A variante do problema FCNDP denominada em [11] por FCNDP-SPR, determina que cada produto $k \in K$ seja transportado por uma rota ótima entre a sua origem o^k e o seu destino d^k. Esta rota ótima é o CMC entre o^k e d^k em relação aos custos c_a. Essa característica adiciona novas restrições ao problema geral [11]. No FCNDP-SPR, além de selecionar um subconjunto de arestas de E para ser aberto e encontrar, para cada mercadoria em K, um fluxo admissível na rede resultante de tal modo que a soma dos custos fixos mais os custos do fluxo variável seja minimizado, cada produto k em K deve ser transportado por um caminho ótimo entre o^k e d^k, no que diz respeito ao comprimento c_a dos arcos utilizados. O FCNDP-SPR pode ser modelado como um problema 0-1 BLPP [11].

Como explicado em [11], o FCNDP-SPR envolve duas tomadas de decisão por parte de dois agentes distintos agindo não cooperativamente e de uma forma sequencial. No nível superior, o líder é responsável por escolher quais as arestas que serão abertas, com o objetivo de minimizar os custos fixos e variáveis. Como resposta, no nível inferior os seguidores devem escolher um conjunto de caminhos mais curtos na rede resultante, pelos quais os produtos serão enviados. No fundo, o problema do nível inferior define um conjunto de problemas independentes de CMC na rede resultante das decisões tomadas no nível superior. O efeito de cada decisão que é tomada sobre a outra parte é indireto: a decisão dos seguidores é afetada pela escolha das arestas feita inicialmente pelo líder. Em contrapartida, a escolha do líder é afetada pelos custos decorrentes das rotas definidas pelo seguidor. No fundo, ambos são influenciados, mas nenhum age diretamente sobre o outro.
No caso do exemplo da Figura 3.2, apenas iríamos escolher da rede projetada, os caminhos mínimos entre as origens e os destinos, pelo que a solução admissível do FCNDP-SPR para o par $o_1 - d_1$ é o caminho $P_1 : 1 - 2 - 4$ e a solução admissível para enviar a segunda mercadoria é o caminho $P_2 : 2 - 4 - 5$. Isto porque estamos neste caso a considerar o menor comprimento dos arcos. Para calcular o custo destas soluções para o líder, tínhamos de calcular o custo de abrir as arestas utilizadas, que neste caso seria 9. Para além disso, transportar a mercadoria 1 custa 8 de acordo com o caminho referido atrás e para a mercadoria 2 custa 4. Logo, o custo total seria 21.

Apesar da sua importância, existe na literatura pouco material acerca do FCNDP-SPR. O FCNDP-SPR foi abordado poucas vezes na literatura [14, 5] e foi tratado como parte de grandes problemas em algumas aplicações [12]. Alguns trabalhos podem ser encontrados em [11, 14, 5].

Em [5] o problema aparece no planeamento de tráfico de redes de transporte urbano onde os utilizadores são assumidos para escolher o CMC quando viajam de um ponto para o outro, na cidade. Uma abordagem heurística é proposta para resolver um exemplo real com 68 vértices e 476 arestas, mas o roteamento do CMC não é formulado explicitamente.

No trabalho mais recente [14], o problema FCNDP-SPR aparece no desenho de uma rede rodoviária para o transporte de materiais perigosos. Na solução do presente problema, o governo define uma seleção de segmentos de estrada a serem abertos ou fechados para o transporte de materiais perigosos, assumindo que os embarques de materiais perigosos na rede resultante serão feitos ao longo de caminhos mais curtos. Este é um caso particular do FCNDP-SPR visto que não há custos associados à seleção de estradas para compor a rede. O governo pretende minimizar a exposição da população no caso de um acidente durante o transporte de mercadorias perigosas. A formulação de programação linear em dois níveis é usada para descrever a relação entre os tomadores de decisão. A solução de um exemplo real é discutida para 48 vértices e 57 arestas.

O FCNDP-SPR também foi tratado como parte de problemas em algumas aplicações maiores. Em telecomunicações, por exemplo, as restrições de CMC são utilizadas para modelar o roteamento do tráfico de procura, no contexto de projeto de redes de protocolos da *internet*.

3.1 Modelo Inteiro Linear em Dois Níveis

Vamos agora descrever a modelação do FCNDP-SPR como um 0-1 BLPP, apresentada em [11].

Note que cada produto $k \in K$ deve ser enviado ao longo de um único caminho ligando a sua origem ao seu destino. Desta forma, para cada produto $k \in K$ será definido um vetor de procura $b^k \in \{-1, 0, 1\}^{|V|}$ especificado como se segue:

$$b_i^k = \begin{cases} -1, & \text{se } i = d^k, \\ 1, & \text{se } i = o^k, \\ 0, & \text{caso contrário.} \end{cases}$$

Para cada aresta $e \in E$, definimos uma variável de nível superior $y_e \in \{0, 1\}$, que é a variável de decisão do líder, tal que $y_e = 1$ apenas se a aresta e é parte do desenho da rede. Então, para cada $e \in E$ definimos:

$$y_e = \begin{cases} 1, & \text{se a aresta } e \text{ pertence à solução,} \\ 0, & \text{se a aresta } e \text{ não pertence à solução.} \end{cases}$$

Também, para cada produto $k \in K$ definimos um conjunto de variáveis de nível inferior x_a^k, $a \in A^E$, usadas para modelar o CMC do vértice o^k para d^k, tendo em conta os comprimentos c_a.

$$x_a^k = \begin{cases} 1, & \text{se na solução o produto } k \text{ passa pelo arco } a, \\ 0, & \text{se na solução o produto } k \text{ não passa pelo arco } a. \end{cases}$$

Então, o problema FCNDP-SPR pode ser formulado como um 0-1 BLPP da seguinte maneira:

$$\min_y \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A^E} g_a^k x_a^k \tag{3.1}$$

$$y_e \in \{0, 1\}, \qquad \forall e \in E, \tag{3.2}$$

onde x é solução ótima para o problema

$$\min_x \sum_{k \in K} \sum_{a \in A^E} c_a x_a^k \tag{3.3}$$

Sujeito a

$$\sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b_i^k, \qquad \forall i \in V, \forall k \in K, \tag{3.4}$$

$$x_a^k \leq y_{e(a)}, \qquad \forall a \in A^E, \forall k \in K, \tag{3.5}$$

$$x_a^k \geq 0, \qquad \forall a \in A^E, \forall k \in K. \tag{3.6}$$

Em (3.4) $\delta^+(i)$ denota o conjunto dos arcos que saem do vértice i e $\delta^-(i)$ denota o conjunto dos arcos que chegam ao vértice i.
A função objetivo do líder, representada por (3.1) pretende minimizar o risco total da rede utilizada pelo seguidor (minimizar o somatório dos custos fixos e variáveis), enquanto que o objetivo do seguidor (3.3) é minimizar o custo (minimiza o somatório das distâncias percorridas no transporte dos produtos). As restrições (3.2) garantem que y_e assume apenas valores binários. As bem conhecidas restrições de conservação de fluxo (3.4) (ver em [1]), asseguram o fluxo da mercadoria k desde a sua origem até ao seu destino. As restrições (3.5) asseguram que apenas os

arcos selecionados pelo governo podem ser usados pelas transportadoras, impondo que não haja fluxo em arcos em que as arestas correspondentes estão fechadas. Observe-se que, uma vez que cada variável y_e é considerada uma constante no problema do nível inferior, a matriz associada às suas restrições é unimodular. Consequentemente, não há necessidade de impor que as variáveis x_a sejam inteiras. Este problema é semelhante ao primeiro modelo de dois níveis proposto por *Kara* e *Verter* [14].

De realçar também que após a definição da rede feita pelo líder, não existe nenhuma restrição que impeça a existência de múltiplos caminhos mínimos para um ou mais produtos. O seguidor poderá nessa situação escolher qualquer um dos caminhos mínimos disponíveis, independentemente do custo associado a cada um deles pelo líder. Contudo, vamos trabalhar com a solução otimista, em que no caso de existirem múltiplos caminhos mínimos, o seguidor escolhe o que representa o menor custo para a função objetivo do líder. Conclui-se portanto que este é um problema 0-1 BLPP, onde o líder e o seguidor interagem indiretamente. A solução otimista representa um cenário onde o seguidor deseja colaborar com o líder e esta é a versão modelada neste trabalho. Alternativamente, poderíamos estar interessados numa solução pessimista, onde cada vez que um dado produto puder ser enviado por dois ou mais caminhos mínimos, será escolhido o que representar o maior custo para a função objetivo do líder. Esta solução representa um cenário onde o seguidor pretende atrapalhar o líder. Sendo assim, dada a rede definida pelo líder, a solução pessimista, dar-nos-á a pior solução possível. Ainda existe uma terceira abordagem possível, denominada solução estável, em que as redes onde existem múltiplos caminhos mínimos são descartadas, optando por aquelas em que o seguidor tem apenas uma escolha possível, mesmo que essa opção leve a uma solução mais custosa para o líder. Nesta solução, o líder não pode dar margem de escolha ao seguidor. A modelação apresentada em [2] segue a abordagem pessimista.

Note que resolver o problema linear definido de (3.3) a (3.6) é equivalente a resolver $|K|$ problemas independentes de caminhos mais curtos, definidos sobre o grafo orientado induzido pelas variáveis y, definido como,

$$D^y = (V; A^y), A^y = \{a \in A^E | y_{e(a)} = 1\}.$$

Uma última observação sobre a formulação de dois níveis é que, uma vez que todos os custos de viagem c_a são positivos, só podemos usar um arco numa direção (uma vez que queremos o caminho mínimo não vamos querer usar o arco para voltar para trás). Assim, podemos substituir as restrições (3.5) pelas seguintes:

$$x^k_{a(e)} + x^k_{\bar{a}(e)} \leq y_e, \qquad \forall e = [i,j] \in E, \ \ \forall k \in K \tag{3.7}$$

Para o modelo de dois níveis isto não nos daria nenhuma vantagem, mas não é o caso para a reformulação proposta de seguida como um modelo linear inteiro em um nível.

3.2 Modelo Inteiro Linear em Um Nível

O modelo (3.1)-(3.6) pode ser transformado num modelo inteiro linear em um nível. Vamos de seguida apresentar a formulação do problema FCNDP-SPR em um nível, conforme apresentado em [11]. Substituindo o programa linear do nível inferior definido por (3.3), (3.4), (3.6) e (3.7) pelas suas condições de otimalidade, podemos reescrever o problema FCNDP-SPR como um problema de PLI num único nível. Inicialmente, aplicando o teorema fundamental da dualidade e o teorema das folgas complementares [3], obtém-se a formulação não-linear:

$$\min_{y,x} \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A^E} g^k_a x^k_a$$

Sujeito a: (3.2), (3.4), (3.6), (3.7),

$$\pi_i^k - \pi_j^k - \lambda_{e(a)}^k \leq c_a, \qquad \forall a \in A^E, \ \forall k \in K, \tag{3.8}$$

$$(y_e - x_{a(e)}^k - x_{\overline{a}(e)}^k)\lambda_e^k = 0, \qquad \forall e \in E, \ \forall k \in K, \tag{3.9}$$

$$(c_a - \pi_i^k + \pi_j^k + \lambda_{e(a)}^k)x_a^k = 0, \quad \forall a = (i,j) \in A^E, \ \forall k \in K, \tag{3.10}$$

$$\lambda_e^k \geq 0, \qquad \forall e \in E, \ \forall k \in K. \tag{3.11}$$

As variáveis π_i^k e λ_e^k são variáveis duais associadas, respetivamente, com restrições em (3.4) e (3.7) definidas para $i \in V$, $e \in E$ e $k \in K$. As restrições (3.8) são as restrições duais associados às variáveis primais x_a^k no problema do nível inferior, ao passo que as restrições não lineares (3.9) e (3.10) são restrições de folga complementares escritas para o par de problemas lineares primal-dual. Tendo em conta o fato de que as variáveis x_a^k e y_e assumem valores inteiros na solução ótima, podemos linearizar estas restrições não-lineares introduzindo no problema uma constante M^k adequadamente grande, para cada mercadoria em $k \in K$. Esta linearização é obtida trocando (3.9) e (3.10), respetivamente por:

$$\lambda_e^k + M^k(y_e - x_{\overline{a}(e)}^k - x_{a(e)}^k) \leq M^k, \qquad \forall e \in E, \ \forall k \in K, \tag{3.12}$$

$$M^k x_a^k - \pi_i^k + \pi_j^k + \lambda_{e(a)}^k \leq M^k - c_a, \qquad \forall a = (i,j) \in A^E, \ \forall k \in K, \tag{3.13}$$

e adicionando:

$$x_a^k \in \{0,1\}, \qquad \forall a \in A^E, \ \forall k \in K. \tag{3.14}$$

A constante M^k, $k \in K$, tem que ser suficientemente grande para garantir que as restrições (3.9) e (3.10) são satisfeitas.
Assim, a formulação linear inteira em um nível é a seguinte:

$$\min_{y,x} \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A^E} g_a^k x_a^k$$

sujeito a (3.2), (3.4), (3.7), (3.8), (3.11)-(3.14).

O problema do seguidor é na realidade um conjunto de diversos problemas de CMC. Assim, as condições de otimalidade do problema do seguidor, são condições de otimalidade de problemas de caminho mínimo, que podem ser escritas de uma forma mais compacta, considerando as condições de otimalidade de *Bellman* [1] para o problema de CMC e utilizando um processo simples de *lifting*. Assumindo a integralidade da variáveis y e x e assumindo também que as desigualdades (3.7) são satisfeitas, as restrições seguintes são equivalentes às condições de otimalidade *Bellman* para o conjunto de $|K|$ pares OD no grafo D^y.

$$\pi_i^k - \pi_j^k \leq M^k - y_{e(a)}(M^k - c_a) - 2c_a x_a^k, \qquad \forall a \in A^E, \ \forall k \in K. \tag{3.15}$$

Considere um arco $a = (i,j) \in A^E$. De acordo com a definição do problema FCNDP-SPR, $y_{e(a)} = 1$ significa que a aresta $e(a)$ foi aberta e isso implica que, para qualquer mercadoria $k \in K$, o arco a pode ser usado no caminho mínimo de o^k para d^k. Assim, se $y_{e(a)} = 1$ e $x_a^k = 1$, então o arco a pertence ao caminho mínimo. Neste caso, a restrição (3.7) implica que $x_{\overline{a}}^k = 0$ e as restrições (3.15) escritas para ambos os arcos a e $\overline{a}$, implicam as condições de *Bellman* satisfeitas com igualdade:

$$\pi_i^k - \pi_j^k = c_a \tag{3.16}$$

Se $y_{e(a)} = 1$ e $x_a^k = 0$ então o arco a não pertence ao caminho mínimo, mas as condições de otimalidade de *Bellman*

$$\pi_i^k - \pi_j^k \leq c_a \tag{3.17}$$

têm de ser satisfeitas para este arco e segue-se a partir de (3.15) para a escrita de a.

Por outro lado, na solução do problema FCNDP-SPR, $y_{e(a)} = 0$ significa que a aresta $e(a)$ não foi aberta e os arcos a e $\overline{a}$ não podem ser usados em nenhum caminho mínimo. Neste caso, as restrições (3.7) escritas para e implicam que:

$$x_{\overline{a}}^k = 0 \text{ e } x_a^k = 0$$

e o valor constante M^k torna as restrições (3.15) escritas para a e $\overline{a}$ redundantes.
Segue a formulação de um nível mais compacta:

$$\min_y \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A^E} g_a^k x_a^k$$
$$y_e \in \{0,1\}, \quad \forall e \in E,$$

Sujeito a: (3.4), (3.7), (3.15)

$$\begin{aligned} & x_a^k \in \{0,1\}, && \forall a \in A^E, \ \forall k \in K, \\ & \pi_{d^k}^k = 0, && \forall k \in K, \\ & \pi_i^k \geq 0, && \forall i \in V, \ \forall k \in K. \end{aligned} \tag{3.18}$$

Note-se que, as variáveis π_i^k, $k \in K$, $i \in V$, são as distâncias mais curtas dos vértices $i \in V$ para o vértice d^k. De seguida, podemos igualar as variáveis $\pi_{d^k}^k$ a zero.

Capítulo 4

Algoritmo Exato para o 0-1 BLPP

Como foi visto no capítulo anterior, o FCNDP-SPR pode ser formulado como um 0-1 BLPP. Portanto, vamos mostrar agora um algoritmo para resolver de forma exata um 0-1 BLPP. Esse algoritmo vem descrito no artigo *'An algorithm for the Discrete Bilevel Programming Problem'* de *Jonathan F. Bard* e *James T. Moore* [17]. De referir que todo o texto deste capítulo é baseado neste artigo.

Na próxima seção, é apresentada notação para o jogo líder-seguidor, usada neste capítulo e também uma reformulação deste jogo como um problema de PLI paramétrica cujos parâmetros aparecem do lado direito das restrições. Isto faz com que tenhamos um problema diferente de cada vez que os parâmetros se alteram, pois o problema do seguidor tornar-se-á dependente destes parâmetros. Este novo problema será usado como base do algoritmo apresentado de seguida. Resolvemos também um exemplo posteriormente, para mostrar o funcionamento do algoritmo.

4.1 Notação e reformulação do problema

Apresentamos de seguida a reformulação linear do 0-1 BLPP como um problema paramétrico. Vamos assumir o seguinte formato para o problema 0-1 BLPP:

$$maximizar_y F(y,x) = c^1 y + c^2 x, \tag{4.1}$$

$$y \in Y = \{y_j \text{ binário}; j = 1, \ldots, n_1\} \tag{4.2}$$

onde x resolve

$$maximizar_x f(y,x) = d^1 y + d^2 x, \tag{4.3}$$

Sujeito a

$$g(y,x) = Ay + Bx \leq b, \tag{4.4}$$

$$x \in X = \{x_j \text{ binário}; j = 1, \ldots, n_2.\} \tag{4.5}$$

Começamos por converter a função objetivo do líder numa restrição parametrizada, e de seguida tentamos resolver o problema resultante. Isso produz uma solução candidata que é usada para encontrar um ponto na região admissível do 0-1 BLPP. Como já mencionado, será assumido que tanto o líder como o seguidor, têm conhecimento total da função objetivo do outro, mas a cooperação não é permitida. Isto implica que as soluções para o 0-1 BLPP podem ser dominadas, mas não necessariamente Pareto-ótimas.

Apresentamos de seguida a notação que será usada ao longo do texto.

A região do espaço definida pelas variáveis de decisão, delimitada pelas restrições, e onde no seu interior ou na fronteira se localiza o máximo ou o mínimo da função objetivo é denominada **Região de Restrições** do 0-1 BLPP. Este conjunto determina uma região onde o ponto ótimo deve estar contido.

$$\Omega = \{(y, x) : Ay + Bx \leq b,\ x \in X, y \in Y\} \tag{4.6}$$

As decisões que o seguidor pode tomar depois do líder ter escolhido y, encontram-se na chamada **Região Admissível** do Seguidor definido para $y \in Y$ fixo. Esta região é dada por:

$$\Omega(y) = \{x \in X : Ay + Bx \leq b\} \tag{4.7}$$

Definimos também o **Conjunto de Reações Racionais** do seguidor como o conjunto das melhores respostas possíveis que o seguidor tem para oferecer após a escolha do líder, da seguinte forma:

$$\Psi(y) = \{x : x \text{ resolve } Max\{f(y, x) : x \in \Omega(y)\}\} \tag{4.8}$$

Por fim, definimos a **Região induzível** IR, que representa o conjunto através do qual o líder pode otimizar caso lhe dêem o total controlo de todas as variáveis, como:

$$IR = \{(y, x) : y \in Y, x \in \Psi(y)\}. \tag{4.9}$$

Para se garantir que o jogo é bem definido, impomos duas condições adicionais:

- Ω é não vazio, pois tem de existir alguém para jogar (tem de existir um par $(y, x) \in \Omega$),
- para cada decisão do líder, o seguidor tem uma resposta admissível; ou seja, $\Omega(y)$ é diferente de $\emptyset$.

O 0-1 BLPP pode então ser escrito como:

$$Max(F(y, x) : (y, x) \in IR). \tag{4.10}$$

Se $\Psi(y)$ é um conjunto não unitário no ótimo (chame-lhe $y*$), valores diferentes de x em $\Psi(y*)$ podem produzir valores diferentes da função objetivo para o líder, mas apenas um subconjunto destes valores maximizam $F(y*, x)$.

O algoritmo que vamos estudar neste capítulo assume que o $\Psi(y*)$ é um conjunto com apenas um elemento ou então, se o seguidor tiver dois caminhos para o mínimo, vai escolher o conjunto das reações racionais que maximiza $F(y*, x)$, isto é, o que é melhor para o líder. Este último caso representa um cenário otimista como assumido no capítulo 3.

A chave do algoritmo está no reconhecimento que qualquer solução para [(2.4)-(2.7)] tem que ter x pertencente ao conjunto de reação racional do seguidor, $\Psi(y)$. Limitando a nossa procura a este conjunto, enquanto tentamos constantemente melhorar a função objetivo do líder, somos capazes de descobrir bons pontos na região induzível, isto é, no conjunto através do qual o líder pode otimizar, caso tenha o total controlo das variáveis. No algoritmo que iremos descrever, isto é conseguido formulando e resolvendo instâncias repetidamente do seguinte programa inteiro parametrizado, derivado de [(2.4)-(2.7)]:

$$\max_{x,y} f(x) = d^2 x, \tag{4.11}$$

Sujeito a

$$Ay + Bx \leq b, \tag{4.12}$$
$$F(y,x) = c^1 y + c^2 x \geq \alpha, \tag{4.13}$$
$$\sum y_j \geq \beta, \tag{4.14}$$
$$x \in X, y \in Y. \tag{4.15}$$

onde α e β são os parâmetros que aparecem do lado direito das restrições. Observe-se que obtemos um problema diferente de cada vez que estes parâmetros mudam. Inicialmente serão tomados como -∞ e 0, respetivamente. A restrição (4.11) obriga a uma troca entre as duas funções objetivo. Por último, a restrição (4.12) restringe o somatório das variáveis controladas pelo líder, e embora não seja estritamente necessário, é uma maneira efetiva de selecionar variáveis para o *branching* neste esquema de enumeração implícita.

4.2 O Algoritmo

Vamos agora apresentar e explicar o algoritmo para resolver o 0-1 BLPP. Note que será mostrado o algoritmo geral, como descrito em [17], ainda sem as alterações necessárias para a adaptação ao nosso problema. Esta adaptação será feita no capítulo 6.
Neste algoritmo é usado um esquema de enumeração implícita, com busca em profundidade centrada nas variáveis de decisão do líder, aplicado ao problema [(4.11)-(4.15)], para resolver [(2.4)-(2.7)]. Assume-se que o leitor está familiarizado com os conceitos associados a árvore de procura binária. A ideia básica é sucessivamente examinar pontos que satisfaçam as restrições do problema [(4.11)-(4.15)], fixar as variáveis y nos seus valores correspondentes, e de seguida voltar a resolver [(4.11)-(4.15)] para obter um novo ponto na região induzível. Ajustando os parâmetros α e β a cada iteração, o algoritmo vai aumentando sucessivamente a função objetivo do líder até que o problema se torne inadmissível. São procurados melhoramentos incrementais, com o objetivo de encontrar um ótimo global.

De forma a fornecer uma estrutura para enumerar as variáveis y, seja $W = \{1, \ldots, n_1\}$ e defina-se na k-ésima iteração do algoritmo o vetor caminho P_k que na árvore de procura é representado por um vetor l-dimensional de tamanho $l = |W_k|$ ($W_k \subseteq W$), onde l é o nível de profundidade da árvore. Aqui $l = |W_k|$ é o comprimento do caminho desde a raiz da árvore até ao nó atual. O conjunto de variáveis de índice atribuído é denotado por W_k. A ordem das componentes em P_k é determinada pelo seu "nível". Isto corresponde a uma atribuição qualquer de $y_j = 0$ ou $y_j = 1$ para $j \in W_k$. Definimos também os seguintes conjuntos:

$$S_k^+ = \{j : j \in W_k \text{ e } y_j = 1\},$$
$$S_k^- = \{j : j \in W_k \text{ e } y_j = 0\},$$
$$S_k^0 = \{j : j \notin W_k\}.$$

O vetor P_k identifica uma solução parcial para as variáveis controladas pelo líder no l-ésimo nível da árvore de procura. Os índices apenas surgem no vetor P_k se estão em S_k^+ ou S_k^-, e aparecem sublinhados se estiverem em S_k^-. Na operação de *branching*, se é selecionada mais do que uma variável, então será anexada ao vetor caminho na ordem ascendente. O vetor caminho P_k indica a ordem em que as variáveis em W_k foram atribuídas, bem como o seu estado binário. Note que cada P_k define um único nó na árvore. Por exemplo, se $y_3 = 0$ e $y_2 = 1$ foram (nesta ordem) fixados na iteração k, então $W_k = \{2, 3\}$, $l = 2$ e $P_k = (\underline{3}, 2)$.
As variáveis em S_k^0 ainda por fixar, são chamadas variáveis livres. A concretização de W_k

é uma atribuição de valores binários às variáveis livres controladas pelo líder. O conjunto das variáveis livres, isto é, não fixadas, constitui o conjunto dos índices S_k^0. O algoritmo nunca coloca explicitamente restrições nas variáveis do seguidor. Depois de k iterações, seja IR^k o conjunto de pontos na região induzível até agora descoberta, e denote $\underline{F}$ o limite inferior associado à função objetivo do líder, definido por, $\underline{F} = max\{F(y,x) : (y,x) \in IR^k\}$. No início do algoritmo, todas as variáveis são livres e $\underline{F} = -\infty$.

4.2.1 Algoritmo passo a passo

Os passos do algoritmo estão descritos no Algoritmo 1 e vamos explicá-los de seguida. Começamos por fazer a inicialização nas linhas 1 e 2, representando o primeiro nó da árvore de procura. Este primeiro nó é um ponto que marca o início do algoritmo, uma vez que para começar ainda não temos nenhuma solução escolhida. Ao criar a raiz da árvore, temos $k = 0$. E cada vez que é adicionado um nó na árvore, é incrementado k. Por isso, o algoritmo será repetido enquanto existirem nós a pesquisar na árvore, o que significa que será enquanto k for maior ou igual a zero. Começamos por criar o nó 0, quando fazemos a inicialização na linha 2. Este nó é a raiz da árvore em que $S_0^+ = S_0^- = \emptyset$, $S_0^0 = \{1, ..., n_1\}$, como pretendemos maximizar a função, α bem como $\underline{F}$, terão um valor muito pequeno e $\beta = 0$.

Após a inicialização, quando entramos na repetição da linha 4, começamos por procurar na linha 5 uma nova solução que é potencialmente admissível nos dois níveis do problema através da solução do problema paramétrico. No fundo escolhemos um nó da árvore ainda não investigado, para ser estudado. Estamos portanto a considerar fixar algumas variáveis em 1 e outras em 0. O contador k apenas é incrementado se a procura tiver sucesso, o que significa que k é incrementado cada vez que é feito o *branching* (quando adicionamos um nó na árvore). Seja (y^k, x^k) a solução encontrada para o problema paramétrico. De seguida na linha 9, fixamos y em y^k, as restrições (4.11) e (4.12) são relaxadas, e o sub-problema resultante é resolvido para encontrar um ponto na região induzível IR^k. Cada vez que é encontrado um melhoramento, o limite inferior é substituído na linha 10. Qualquer solução admissível não visitada encontrada neste ponto será acompanhada por $J \neq \emptyset$.

O conjunto J é o conjunto das variáveis de *branching*. Farão parte de J todas as variáveis que não estavam fixas nem em 1, nem em 0 e que acabámos de fixar em 1. Se $J = \emptyset$, actualiza-se o β que será igual ao número de elementos de S_k^+ acrescido de uma unidade. Na linha 23, o algoritmo retorna então ao primeiro passo depois da inicialização para resolver novamente o mesmo problema, com novos parâmetros. [1]

O *branching* ocorre nas linhas 15 até à 18 onde S_k^+, S_k^0 e P_k são atualizados. No processo, todas as variáveis livres iguais a 1 na solução da linha 5 são fixadas a 1. São elas que compõem o conjunto J, conjunto esse que por sua vez vai determinar as variáveis de *branching*, uma vez que cada variável em J gera um novo nó no *branching*, sendo criado na árvore de procura o correspondente número de novos nós. Contrariamente ao procedimento de busca em profundidade habitual, onde um novo nó é criado em cada iteração, aqui encontramos vantagens em estender a árvore vários níveis de cada vez.

Na linha 19, α é definido como o melhor limite inferior de [(2.4)-(2.7)] conhecido até ao momento $\underline{F}$, incrementado de uma unidade. Isto garante que quando o algoritmo volta à linha 5, se uma solução admissível de [(4.11)-(4.15)] é encontrada, (y^k, x^k), satisfará $F(y^k, x^k) > \underline{F}$. Também na linha 19, β é definido como o somatório das variáveis controladas pelo líder e é incrementado de uma unidade. Isto garante que no mínimo um elemento do conjunto S_k^0 na

[1] Isso pode acontecer quando no *backtracking* fixamos uma variável a zero e fazemos $\beta = 0$.

Algoritmo 1 Algoritmo de enumeração implícita para o 0-1 BLPP

Inicialização: cria raiz da árvore de procura
$k = 0;\ S_0^+ = \emptyset;\ S_0^- = \emptyset;\ S_0^0 = \{1, \ldots, n_1\};\ \alpha = -\infty;\ \beta = 0;\ \underline{F} = -\infty$
while $k \geq 0$ **do**
  $y_j = 1$ para $j \in S_k^+$; $y_j = 0$ para $j \in S_k^-$;
  Para os valores atuais de α e β encontrar solução admissível para [(4.11)-(4.15)]
  if encontrou solução admissível para [(4.11)-(4.15)] **then**
    Nomear a solução obtida como (y^k, x^k)
    //*bounding* (limites)
    Fixar y em y^k e resolver problema seguidor para encontrar $(y^k, \hat{x}^k) \in IR$ que torna mínimo o custo da função objetivo do seguidor - $F(y^k, \hat{x}^k)$
    Calcular $\underline{F} = max\{\underline{F}, F(y^k, \hat{x}^k)\}$
    $//J$ = variáveis que não estavam fixas e fixámos em 1
    $J = \{j \in S_k^0 \text{ e } y_j^k = 1\}$
    if $J \neq \emptyset$ **then**
      //*branching* (ramificação)
      backtracking=0;
      Incrementar k de cada vez que criar um nó
      Criar $|J|$ novos nós: juntar os elementos $j \in J$ a P_{k-1} na ordem ascendente, 1 a 1, para obter os novos nós disponíveis e o caminho final P_k
      $S_k^+ = S_{k-1}^+ \bigcup J;\ S_k^0 = S_{k-1}^0 \setminus J;\ S_k^- = S_{k-1}^-;$
      $\alpha = \underline{F} + 1; \beta = 1 + |S_k^+|;$
    else
      $S_k^+ = S_{k-1}^+;\ S_k^- = S_{k-1}^-;\ S_k^0 = S_{k-1}^0;\ P_k = P_{k-1};$
      $\alpha = \underline{F} + 1;\ \beta = 1 + |S_k^+|;$
      Voltar ao início do while para resolver o mesmo problema com novos parâmetros.
    end if
  else
    Descartar nó atual e realizar *backtracking*
    backtracking=1;
  end if

  // *backtracking*
  if (backtracking==1) **then**
    while $k \geq 0$ **do**
      Voltar atrás desde o nó atual 1 nível (nó mais recentemente criado)
      if $(y_{j'}^k = 1)$ **then**
        //Ramificar o seu complementar (Chamar a variável correspondente j')
        $k = k + 1;$
        $y_{j'}^k = 0;$
        Atualizar S_k^+, S_k^-, S_k^0, P_k, de acordo com as suas definições
        $\beta = 0;$
        continue;
      end if
    end while
  end if
  if não existem mais nós **then**
    if F=valor muito grande **then**
      não existe solução admissível para [(2.4)-(2.7)]
    else
      declarar ponto admissível associado a solução ótima F
    end if
    break;
  end if
end while

iteração k tem o valor de 1, fornecendo assim uma solução ainda não visitada quando retornar à linha 5. Se a restrição (4.12) não for incluída, os mesmos valores de y podem resultar quando em [(4.11)-(4.15)] é depois resolvido na linha 5. Caso isto ocorra, diz-se que o algoritmo estagna porque nenhum progresso é possível sem a introdução de novas regras de ramificação. Incluímos esta restrição para evitar gerar a mesma solução [2].

Se o problema [(4.11)-(4.15)] é inadmissível na linha 5, o algoritmo procede para o *backtracking* na linha 26 onde a operação de *backtracking* na árvore de busca é iniciada. Note que o nó aberto (pois ainda está em exploração) é o que está associado ao subproblema definido pela combinação de elementos em S_k^+ e S_k^- que ainda tem de ser plenamente explorado ou *fathomed*. Como o algoritmo começa por ramificar sempre para o lado esquerdo ($y_j = 1$), o *backtracking* é conseguido encontrando o elemento de P_k não sublinhado mais à direita, sublinhando-o e apagando todas as entradas à direita. Embora não tenha sido dito explicitamente no *backtracking*, um novo nó é adicionado à árvore de procura cada vez que é realizada a operação de *backtracking*. A nova entrada sublinhada é apagada de S_k^+ e adicionada a S_k^-; as entradas *erased* são apagadas de S_k^- e adicionadas a S_k^0.

Quando $k = -1$, então significa que o *backtracking* foi feito de forma a eliminar todos os nós da árvore a investigar.

Os autores do artigo [17] consideram que não vale a pena muito esforço para resolver o problema paramétrico de forma ótima. Uma resolução heurística pode ser suficiente na maior parte dos casos. No caso da heurística não encontrar solução, aí sim, seria necessário resolver o problema de forma ótima.

4.3 Exemplo

Neste tópico vai ser apresentada a resolução completa de um exemplo simples, para elucidar os passos do algoritmo. Este exemplo aparece resolvido parcialmente no artigo [17] que propõe o algoritmo aqui descrito. Considere o seguinte problema linear em dois níveis com variáveis contínuas.

$$\max_{y \geq 0} F(y, x) = -y - x,$$

onde x resolve

$$\max_{x \geq 0} f(y, x) = 5y + x,$$

Sujeito a

$$\begin{aligned} -y - x/2 &\leq -2, \\ -y/4 + x &\leq 2, \\ y + x/2 &\leq 8, \\ y - 2x &\leq 4. \end{aligned}$$

A solução ótima deste problema (sem restrições de integralidade) é $(y*, x*) = (\frac{8}{9}, \frac{20}{9})$, com $F = \frac{-28}{9}$. O gráfico da região admissível Ω na Figura 4.1, indica que $y < 8$ e $x < 4$.

[2] No projeto original estudado no artigo [17], (4.12) não aparecia, por isso era necessário desenvolver um procedimento para selecionar a variável *branching* ou de ramificação. Após testes consideráveis, a regra que mostrou ser a mais promissora envolve encontrar o max $\{c_j^1 : j \in S_{k-1}^0 \text{ e } y_j^k \neq 1\}$ para obter o correspondente índice j. Contudo, nem esta regra nem outras regras investigadas funcionaram tão bem como a efetivamente aplicada (usando a restrição (4.12)).

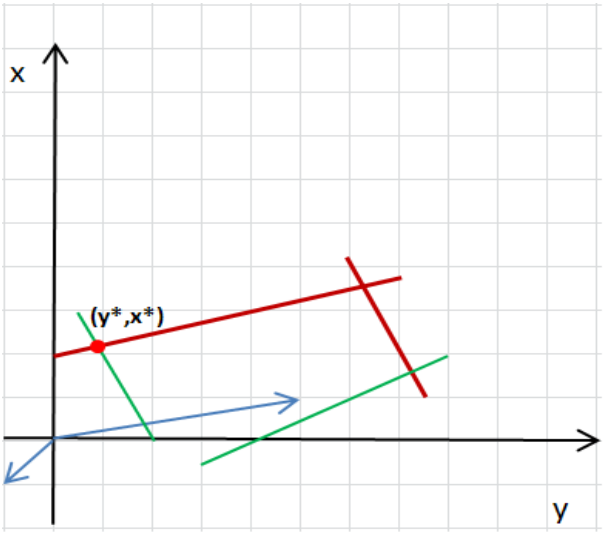

Figura 4.1: Representação gráfica da região admissível.

Alternativamente, a terceira restrição do problema interior mostra que $y <= 8$ (com $y = 8$ requerendo que $x = 0$, o que a 4^a restrição proíbe), enquanto que quatro vezes a segunda restrição mais a 3^a restrição dá $4.5x \leq 16$, ou $x < 4$. Por isso, se x e y estão agora restringidos a valores inteiros, uma formulação 0-1 pode ser obtida tomando $y = y_1 + 2y_2 + 4y_3$ e $x = x_1 + 2x_2$. Isto leva à formulação equivalente mostrada a seguir.

$$\max_{y \in Y} F(y, x) = -y_1 - 2y_2 - 4y_3 - x_1 - 2x_2,$$

onde x resolve

$$\max_{x \in X} f(y, x) = 5y_1 + 10y_2 + 20y_3 + x_1 + 2x_2,$$

Sujeito a

$$\begin{aligned} -2y_1 - 4y_2 - 8y_3 - x_1 - 2x_2 &\leq -4, \\ -y_1 - 2y_2 - 4y_3 + 4x_1 + 8x_2 &\leq 8, \\ 2y_1 + 4y_2 + 8y_3 + x_1 + 2x_2 &\leq 16, \\ y_1 + 2y_2 + 4y_3 - 2x_1 - 4x_2 &\leq 4. \end{aligned}$$

Para resolver este problema, usando o algoritmo que foi aqui descrito, necessitamos de o reescrever na forma paramétrica de [(4.11)-(4.15)]:

$$\max_{x \in X, y \in Y} f(x) = x_1 + 2x_2,$$

Sujeito a

$$\begin{aligned} -2y_1 - 4y_2 - 8y_3 - x_1 - 2x_2 &\leq -4, \\ -y_1 - 2y_2 - 4y_3 + 4x_1 + 8x_2 &\leq 8, \\ 2y_1 + 4y_2 + 8y_3 + x_1 + 2x_2 &\leq 16, \\ y_1 + 2y_2 + 4y_3 - 2x_1 - 4x_2 &\leq 4, \\ F(y, x) = -y_1 - 2y_2 - 4y_3 - x_1 - 2x_2 &\geq \alpha, \\ y_1 + y_2 + y_3 &\geq \beta. \end{aligned}$$

A execução passo a passo do Algoritmo 1 para o exemplo dado está delineada de seguida. Para ser conciso, a representação inteira dos pontos binários é utilizada na discussão. Note que os resultados intermédios não são únicos, mas dependem da aproximação algorítmica utilizada para resolver os subproblemas nas linhas 5 e 9 do Algoritmo 1. A Figura 4.2 mostra a árvore de procura que será construída neste exemplo.

Começamos por inicializar $S_0^+ = S_0^- = \emptyset$, $S_0^0 = \{1,2,3\}$, $\alpha = -\infty$ e $\beta = 0$. Após a inicialização na linha 2 e da criação do nó 0, o algoritmo encontra na linha 5, o ponto (4,3), o qual é confirmado que está na região induzível no *bounding*, nas linhas 9 e 10. Assim, $\underline{F} = -7$.
Fixámos a variável y_3 a 1, por isso, na criação do conjunto J que ocorre na linha 12, este conjunto passará a ter um elemento $J = \{3\}$. De seguida criamos um novo nó (o nó 1) e actualizamos os conjuntos, $S_1^+ = \{3\}$, $S_1^- = \emptyset$, $S_1^0 = \{1,2\}$, $P_1 = (3)$, $\alpha = -6$ e $\beta = 2$.

Resolvendo o problema [(4.11)-(4.15)] novamente na linha 5 obtém-se o ponto (5,1) como solução admissível. No *bounding*, $(y^2, \hat{x}^2) = (5,3)$, $F(y^2, \hat{x}^2) = -8$, logo $\underline{F}$ mantém-se em -7. Como $J = \{1\}$, pois apenas o x_1 foi fixo a um (e anteriormente não estava fixo), iremos criar um novo nó. Actualizando os conjuntos obtemos $S_2^+ = \{1,3\}$, $S_2^0 = \{2\}$, $S_2^- = \emptyset$, $P_2 = (3,1)$, $\alpha = -6$ e $\beta = 3$.

De volta ao ciclo *while*, na linha 5 é detectado que o problema paramétrico definido para o nó 2 é inadmissível. Na linha 5 nenhuma solução admissível é encontrada, por isso o algoritmo transforma em fantasma o nó 2, vai para o *backtracking* e volta atrás dando $S_3^+ = \{3\}$, $S_3^- = \{1\}$, $S_3^0 = \{2\}$, $P_3 = (3,\underline{1})$, α=-6 e β=0.

Voltando à linha 5, é encontrado o ponto admissível $(y^3, \hat{x}^3)$=(4,2), no *bounding* outra vez devolve $(y^3, \hat{x}^3) = (4,3)$ com $F(y^3, \hat{x}^3) = -7$.
Nas linhas 13 à 17, o fato de $S_3^0 = \{2\}$ e $y_2^3 = 0$ implica que $J = \emptyset$, e por consequência, $S_3^+ = S_2^+ = \{3\}$. O parâmetro β é por isso definido na linha 21 a 1 mais $|S_3^+|$ que é 2 neste caso, e o controlo passa à iteração geral na linha 5.

O subproblema resultante não é admissível por isso o algoritmo elimina o nó atual, que é o 3, salta para a linha 30 e faz *backtracking* criando o nó 4. De volta às atualizações, $S_4^+ = \emptyset$, $S_4^- = \{3\}$ e $S_4^0 = \{1,2\}$, com $\beta = 0$.

Encontramos de seguida a solução (3,1). No *bounding* obtém-se a mesma solução, que faz com que $\underline{F}$ actualize para -4. Criamos dois novos nós, o 5 e o 6, porque $J = \{1,2\}$, actualizamos $S_5^+ = \{2\}$, $S_5^- = \{3\}$, $S_5^0 = \{1\}$. Resolve 6. β vai passar a 3 e por termos actualizado $\underline{F}$, α vai tornar-se -4. Encontramos a solução (2,3) que faz com que $F = -5$. Como o valor é inferior a -4, não actualizamos o $\underline{F}$. Actualizamos $S_6^+ = \{1,2\}$, $S_6^- = \{3\}$, $S_6^0 = \emptyset$, $P_6 = (3,2,1)$.

Desconsideramos o nó 6 uma vez que a solução é inadmissível, faz-se *backtracking* e resolve-se 7. Actualizamos $S_7^+ = \{2\}$, $S_7^- = \{3,1\}$, $S_7^0 = \emptyset$, $P_7 = (3,2,\underline{1})$, $\beta = 0$ e $\alpha = -4$. Retomando a linha 4 e 5, é encontrado o ponto admissível $(y^7, \hat{x}^7) = (2,2)$, sublinhando o 1 uma vez que estamos a analisar a sua variável complementar. No *bounding*, devolve o mesmo par com F$(y^7, \hat{x}^7)$=-4. Obtemos portanto $J = \emptyset$, e podemos então atualizar $S_7^+ = \{2\}$, $S_7^- = \{3,1\}$, $S_7^0 = \emptyset$, $P_7 = (3,2,1)$, $\beta = 2$ e $\alpha = -3$. Com estes valores obtemos uma solução inadmissível. Encontramos de seguida a solução (3,2), onde F actualize para -5. Não actualiza $\underline{F}$ uma vez que este valor é pior.

Descartamos então o nó atual 5 e obtemos a solução admissível $(y^8, \hat{x}^8) = (1,2)$, com $F(y^8, \hat{x}^8) = -3$. Temos então $J = \{1\}$, $S_9^+ = \{1\}$, $S_9^- = \{2,3\}$, $S_9^0 = \emptyset$, $P_9 = (\underline{3},\underline{2},1)$, $\beta = 1$ e $\alpha = -2$. Com estes valores obtemos uma solução inadmissível.

Após descartar o nó atual (nó 9) e atualizar apenas β para 0 em relação aos valores atuais, voltamos a ter uma solução inadmissível no nó 10.

Como não existem mais nós, atingimos desta forma a convergência. A solução ótima para o exemplo é $(y*, x*) = (1, 2)$ com $F(1, 2) = -3$. A árvore correspondente é mostrada na Figura 4.2, onde os valores do F, para além dos nós são os encontrados no *bounding*. Durante a execução, o algoritmo volta à linha 4, 13 vezes numa tentativa de melhorar o limite inferior. Subsequentemente, 7 subproblemas são resolvidos no *bounding* para encontrar novos pontos na região admissível. Finalmente, se o algoritmo alternativo é utilizado e o problema [(4.11)-(4.15)] é completamente resolvido na linha 5, três destes sete subproblemas são eliminados no *bounding*. Segue o desenho da árvore de procura deste exemplo, para uma melhor ilustração:

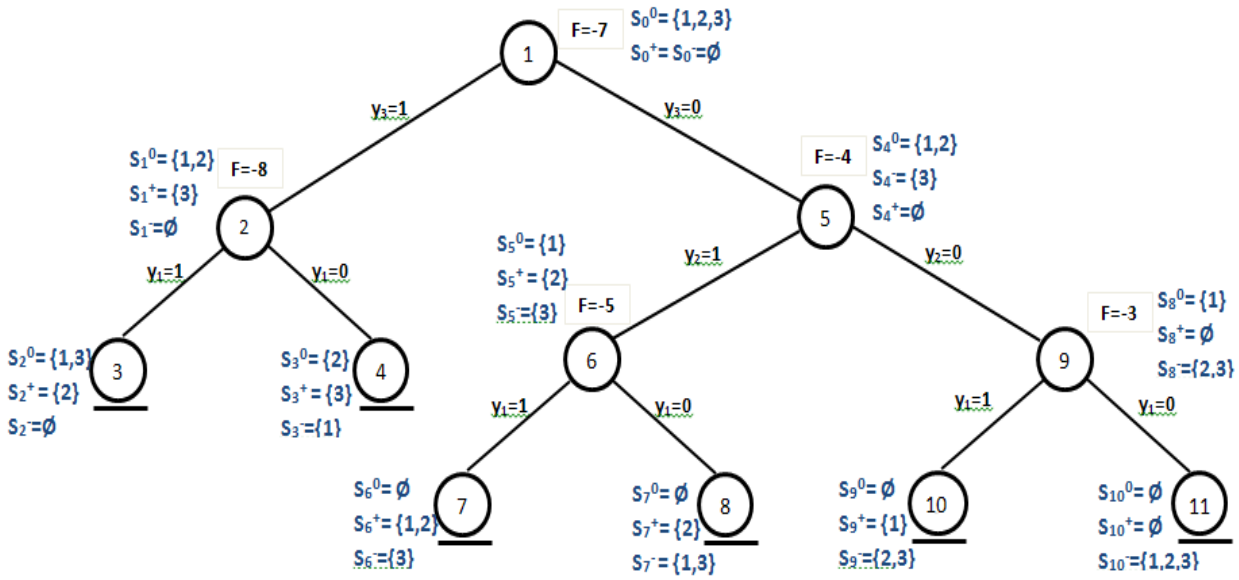

Figura 4.2: Árvore de procura para o exemplo da seção 4.3.

4.4 Correção do Algoritmo

Vamos de seguida apresentar a proposição que nos prova a correção deste algoritmo. Para mais detalhes, ver [17].

PROPOSIÇÃO 1: Seja Y^* o conjunto de soluções ótimas para o líder, assumindo que um ótimo existe. Se $\Psi(y)$ só tem um valor para todo o $y \in Y^*$, ou se $f(x)$ é único para $(y, x) \in \{y \in Y^*, x \in \Psi(y)\}$, então o algoritmo termina com uma solução ótima para o 0-1 BLPP [(2.4)-(2.7)]. Alternativamente, uma condição suficiente para o algoritmo terminar com uma solução ótima é que o seguidor escolha sempre um $x \in \Psi(y)$ que maximiza $F(y, x)$ para y fixo.

Sumariamente, esta proposição mostra-nos que o algoritmo vai funcionar. Supõe-se que ou o conjunto de reação racional do seguidor é unitário ou então iremos trabalhar com a visão otimista. As condições da Proposição 1 não são rapidamente verificáveis, então, a não ser que o subproblema sugerido por *Bard* e *Falk* seja resolvido, não pode ser determinado se o algoritmo termina com o ótimo. Para uma discussão detalhada sobre este tópico, sugerimos ao leitor o artigo original [17]. Os resultados dos testes realizados em [17] indicam que resolvendo a otimalidade de [(4.11)-(4.15)] na linha 5 do algoritmo, oferece poucas vantagens, primeiro porque a

solução (chamada (y^k, x^k)) não está necessariamente na região induzível. Por outro lado, pode existir outro ponto que satisfaça o problema do seguidor, mas que viola o corte (4.11) e fornece o seguidor com um valor maior do que a função objetivo $f(x^k)$.

Capítulo 5

Problemas de Caminho Mais Curto

O problema do CMC é um dos problemas fundamentais da Otimização e é um dos modelos centrais na otimização em redes. Problemas de CMC fazem parte do estudo das redes e fluxos. Estão presentes em situações práticas onde se pretende enviar algum material (como dados empacotados num computador, telecomunicações ou fluxos de veículos) entre dois pontos de uma dada rede, de uma forma rápida, barata e eficaz. São capazes de capturar informação essencial da rede. Servem de ponto de partida para estudos sobre modelos mais complexos, porque são muitas vezes problemas mais pequenos dentro de outros problemas de otimização combinatória, como é o nosso caso. Problemas de CMC podem aparecer em redes onde os comprimentos dos arcos são não negativos ou em redes com comprimentos arbitrários.
Alguns tipos de problemas de CMC são:

1. Encontrar o CMC entre um vértice origem e um vértice destino,
2. Encontrar caminhos mais curtos de um vértice para todos os outros da rede,
3. Encontrar caminhos mais curtos entre todos os pares de vértices do grafo.

Vamos focar-nos no estudo de CMC entre um vértice origem e um vértice destino, dentro de uma determinada rede. Veremos como esse problema é modelado como um problema de PLI e iremos abordar um possível algoritmo para a sua resolução: o Algoritmo de *Dijsktra* [1]. De realçar que para o algoritmo de caminho mínimo funcionar, o grafo tem de ser conexo. Numa segunda fase, será introduzida a noção de restrições adicionais em problemas de CMC, com o exemplo das restrições de capacidade. A inclusão de restrições de caminho mínimo torna o problema mais complicado, tanto na modelação como na busca de soluções eficientes. Será apresentado um algoritmo de programação dinâmica exata, proposto para resolver o problema de CMC com uma restrição deste tipo [18].

No capítulo 6 veremos como os algoritmos de CMC podem ser utilizados na solução do FCNDP-SPR.

5.1 Algoritmos de CMC de Uma Origem para Um Destino

5.1.1 Formulação como problema de programação linear

Considere-se um digrafo $G = (V, A)$ com comprimento do arco (ou custo do arco) c_{ij} associado a cada arco $(i, j) \in A$. O grafo tem um vértice s, chamado de origem. Definimos como sendo o comprimento de um caminho, o somatório de todos os comprimentos de arcos no caminho. O problema do CMC consiste em determinar para um vértice específico t (vértice destino) o CMC desde o vértice s até ao vértice t. No caso de não existir caminho de s para t, diz-se que t não é alcançável a partir de s. De uma forma alternativa, podemos ver o problema como enviando uma unidade de fluxo o mais barato possível (com custos c_{ij} nos arcos) desde o vértice s para o

vértice t, numa rede sem capacidades. Para o problema que queremos resolver, ou seja, para o problema do CMC entre um vértice s e um vértice t, de acordo com esta descrição, a formulação do problema do CMC como problema de PLI fica a seguinte:

$$\text{Minimizar} \sum_{(i,j)\in A} c_{ij}x_{ij}$$

Sujeito a

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ij} = \begin{cases} 1, & i = s, \\ -1, & i = t, \\ 0, & \forall i \in V \setminus \{s,t\}, \end{cases} \tag{5.1}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in A.$$

Modelo 1. Formulação do CMC como problema de PLI de um determinado vértice para outro.

Na formulação do Modelo 1 e supondo que pretendemos enviar uma unidade de fluxo do vértice s para o vértice t, colocamos a igualdade (5.6) a 1, para o vértice s, ou seja, do vértice s sai uma unidade de fluxo e não entra nenhuma. No caso do vértice t, apenas vai entrar a unidade de fluxo, logo a igualdade será -1. Nos restantes vértices, não pretendemos deixar qualquer unidade de fluxo, ou seja, a quantidade que entra no vértice, também sai, daí a igualdade ser 0. Repare que originalmente, precisaríamos definir x_{ij} como 0 ou 1. Entretanto, a matriz A associada ao problema anterior é totalmente unimodular. Essa propriedade garante que a solução dada por x será sempre 0 ou 1, mesmo que seja apenas exigido que $x_{ij} \geq 0$.

Vamos considerar no nosso problema que:

O grafo contém um caminho entre os vértices s e t. Podemos ultrapassar esta limitação adicionando um arco fictício (s,t) com um custo grande, caso s não esteja conectado a t.

O grafo não contém custos negativos, logo não contém ciclos de custo negativo. Para uma rede que contenha um ciclo negativo Q, a formulação do programa linear tem uma solução ilimitada pois enviamos uma quantidade infinita de fluxo ao longo de Q. A situação nestes casos é um pouco diferente: o CMC pode passar num ciclo negativo um número infinito de vezes, sendo que cada repetição reduz o comprimento do passeio. Nestes casos, precisamos proibir o passeio de revisitar tais vértices. Isto é, uma forma de contornar esta situação é impedir a visita a este tipo de ciclo mais do que uma vez. Esta restrição pode trazer implicações computacionais, tornando o problema de CMC mais difícil de resolver.

O grafo é orientado. Se o grafo não for orientado e se todos os comprimentos forem não negativos, podemos transformar este problema de CMC num outro, usando o grafo orientado obtido, bidireccionando cada aresta do grafo não orientado.

5.1.2 Algoritmos *Label-Setting* e Algoritmos *Label-Correcting*

A literatura classifica os algoritmos para resolver problemas de CMC em dois grupos: algoritmos de atribuição de etiquetas *(label-setting)* e algoritmos de correção de etiquetas *(label-correcting)*. Vamos apresentar na seguinte tabela a diferença entre ambos:

	Algoritmos *Label-Setting*	**Algoritmos *Label-Correcting***
Etiquetas	Definem uma etiqueta permanente (ótima) a cada iteração.	Consideram todas as etiquetas temporárias até ao passo final, onde depois se tornam definitivas.
Tipo de grafo	Aplicam-se a problemas de CMC existentes em grafos sem ciclos, com custos arbitrários, e a problemas de CMC sem custos negativos.	São mais gerais e aplicam-se a mais classes de problemas de CMC, incluindo grafos com custos negativos.
Complexidade	São muito mais eficientes, isto é, apresentam uma melhor complexidade de pior-caso.	———
Convergência	As provas de convergência são bastante mais simples e baseiam-se em argumentos elementares de análise combinatória.	As provas de convergência são mais subtis e requerem uma análise mais detalhada.

Tabela 5.1: Comparação entre Algoritmos *Label-Setting* e *Label-Correcting*.

O algoritmo *label-setting* mais básico é o Algoritmo de *Dijkstra* e é o que vamos estudar neste trabalho. Escolhemos este de entre vários, uma vez que no nosso caso não temos custos negativos nos arcos e é o de mais fácil implementação.

5.1.3 Algoritmo *Label-Setting*: Algoritmo *Dijkstra*

No nosso estudo, o algoritmo de *Dijkstra* encontra o CMC desde o vértice origem s até ao vértice destino t num grafo com comprimento de arcos não negativo. Seja $B(i)$ a lista de arcos adjacentes ao vértice i e seja $C = max\{c_{ij} : (i,j) \in A\}$. A distância etiquetada a qualquer vértice permanente representa a distância mais curta desde a origem até esse vértice. Em qualquer passo intermédio, o algoritmo divide os vértices em 2 grupos: os etiquetados permanentemente (ou permanentes) e os temporariamente etiquetados (ou temporários). A ideia base do algoritmo é percorrer os vértices permanentemente etiquetados por ordem das suas distâncias desde o vértice origem s. Inicialmente damos ao vértice s uma etiqueta permanente de valor zero, e a cada um dos outros vértices i, uma etiqueta temporária de valor igual a ∞. Em cada iteração, a etiqueta de um vértice i é a sua distância mais curta conhecida desde o vértice origem s ao longo de um caminho cujos vértices internos (i.e., outros vértices que não i) são todos etiquetados permanentemente. O algoritmo escolhe um vértice i com a mínima etiqueta temporária (quebrando os empates arbitrariamente), torna-a permanente, e recomeça a partir desse vértice. Ou seja, percorre os arcos em $B(i)$ para atualizar as etiquetas de distâncias dos vértices adjacentes ao vértice i. O algoritmo termina quando designar todos os vértices como permanentes. A demonstração da correção do algoritmo reside no fato de que podemos sempre designar o vértice com a mínima etiqueta temporária como permanente. Este algoritmo será descrito no Algoritmo 2. O índice $pred(j)$ regista o antecessor do vértice $j \in V$ no CMC de s a t.

Ilustramos o algoritmo de *Dijkstra* utilizando o grafo da Figura 5.1.

Vamos determinar o CMC de $s = 1$ para $t = 6$. Na tabela 5.2 é apresentada a execução do Algoritmo de *Dijkstra* para o grafo mostrado na Figura 5.1. Da aplicação do algoritmo, concluímos que o CMC de $s = 1$ para $t = 6$ é 1-2-5-6, com custo=6.

Demonstração da Correção do Algoritmo de *Dijkstra*
Usamos argumentos indutivos para estabelecer a validade do algoritmo de *Dijkstra*. A cada ite-

Algoritmo 2 Algoritmo de *Dijkstra*

algoritmo *Dijkstra*;
$S := \emptyset; \overline{S} := N$;
$d(i) := \infty$ para cada vértice $i \in N$
$d(s) := 0$ e $pred(s) := 0$;
while |S| < n **do**
  seja $i \in \overline{S}$ um vértice para o qual $d(i) = min\{d(j) : j \in \overline{S}\}$;
  $S := S \cup \{i\}$;
  $\overline{S}$:= $\overline{S}$ - {i};
  for cada (*i*,*j*) ∈ B(i) **do**
    if d(j) > d(i) + cij **then**
      $d(j) := d(i) + c_{ij}$ e $pred(j) := i$;
    end if
  end for
end while

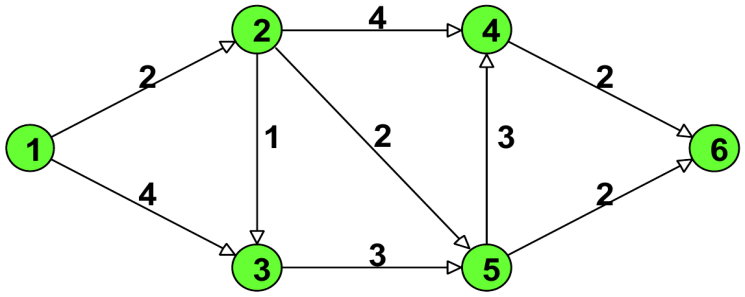

Figura 5.1: Grafo para a aplicação do Algoritmo de *Dijkstra*.

ração, o algoritmo particiona os vértices em 2 conjuntos, S e $\overline{S}$. As nossas hipóteses de indução são:

- Que a etiqueta distância de cada vértice em S é ótima ;
- Que a etiqueta distância de cada vértice em $\overline{S}$ é o caminho com comprimento mais curto desde a origem, admitindo que cada vértice interno no caminho se encontra em S.

Fazemos indução sobre a cardinalidade do conjunto S. Para provar a primeira hipótese de indução, recorde que a cada iteração o algoritmo transfere um vértice i no conjunto $\overline{S}$ com a menor etiqueta de distância para o conjunto S. Precisamos mostrar que a etiqueta de distância $d(i)$ do vértice i é ótima. Pela hipótese de indução, $d(i)$ é o comprimento do CMC para o vértice i, entre todos os caminhos que não contêm nenhum vértice em $\overline{S}$, como interno.

Mostramos agora que o comprimento de qualquer caminho de s para i, que contém alguns vértices de $\overline{S}$ como internos, será pelo menos $d(i)$. Para isso, considere qualquer caminho p da origem para o vértice i, que contém pelo menos um vértice em $\overline{S}$ como interno. O caminho p pode ser decomposto em dois segmentos p_1 e p_2: o segmento de caminho p_1 não contém nenhum vértice em $\overline{S}$ como vértice interno, mas termina no vértice k em $\overline{S}$ (Figura 5.2). Por hipótese de indução, o comprimento do caminho p_1 é pelo menos $d(k)$ e desde o vértice i, é a etiqueta de menor distância em $\overline{S}$, $d(k) \geq d(i)$. Portanto, o segmento de caminho p_1 tem pelo menos comprimento $d(i)$. Além disso, desde que todos os comprimentos dos arcos sejam não negativos, o comprimento do segmento de caminho p_2 é não negativo. Consequentemente, o comprimento do caminho p é pelo menos $d(i)$. Este resultado estabelece o fato que $d(i)$ é o caminho de comprimento mais curto do vértice i a partir do vértice origem.

		Vértices							
Iteração		1	2	3	4	5	6	Temporários	Definitivos
	1	(0,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	(∞,-)	2,3,4,5,6	1
	2		**(2,1)**	(4,1)	(∞,-)	(∞,-)	(∞,-)	3,4,5,6	1,2
	3			**(3,2)**	(6,2)	(4,2)	(∞,-)	4,5,6	1,2,3
	4				(6,2)	**(4,2)**	(∞,-)	4,6	1,2,3,5
	5						**(6,5)**	4	1,2,3,5,6

Tabela 5.2: Resolução do Problema do CMC, aplicando o Algoritmo de *Dijkstra*.

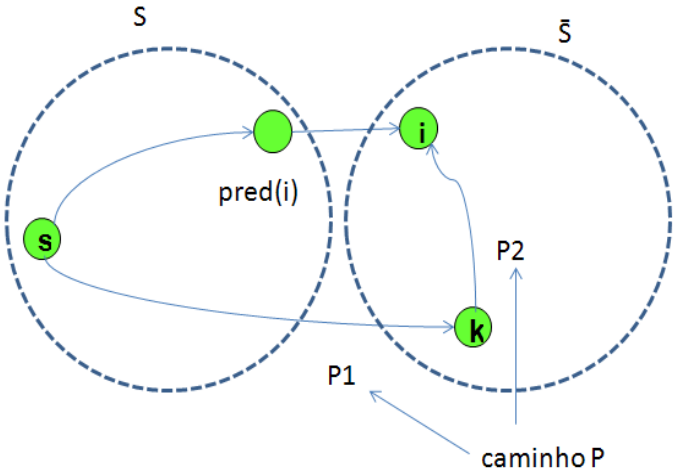

Figura 5.2: Prova do Algoritmo de *Dijkstra*.

De seguida vamos mostrar que o algoritmo preserva a segunda hipótese de indução. Depois do algoritmo ter etiquetado um novo vértice i permanentemente, as etiquetas de distância de alguns vértices em $\overline{S} - i$ devem ser decrementadas, porque o vértice i podia tornar-se um vértice interno na tentativa de caminhos mais curtos para estes vértices. Mas recorde que depois de etiquetar o vértice i permanentemente, o algoritmo examina cada arco $(i, j) \in B(i)$ e se $d(j) > d(i) + c_{ij}$, então configura $d(j) = d(i) + c_{ij}$ e $pred(j) = i$. Portanto, depois de atualizar a distância, pela hipótese de indução, o caminho a partir do vértice j para o vértice origem definido pelos índices predecessores, satisfaz a Propriedade 1, que será decrita a seguir, e assim a etiqueta da distância de cada vértice em $\overline{S} - \{i\}$ é o comprimento do CMC sujeito à restrição de que cada vértice interno no caminho deve pertencer a $S \cup \{i\}$.

Propriedade 1: Seja d o vetor das distâncias mais curtas. Então um caminho orientado p desde o vértice origem até ao vértice k é um CMC se e só se $d(j) = d(i) + c_{ij}$, para cada arco $(i, j) \in p$.

5.2 O problema do CMC elementar com restrições adicionais

Na prática, quando se pretende determinar o CMC entre dois vértices é usual existirem restrições adicionais relativas ao caminho a determinar, nomeadamente, existirem janelas temporais relativas às visitas aos vértices, quando o vértice só pode ser visitado num determinado intervalo de tempo, ou então existirem restrições relativas ao consumo de um recurso. Neste último caso, assume-se que percorrer um arco implica o consumo de um recurso escasso, pelo que o caminho a ser escolhido deve garantir que o consumo do recurso não ultrapassa a disponibilidade máxima

desse recurso. Pode também pretender determinar-se o CMC entre dois pontos de um grafo que passe obrigatoriamente por determinados vértices.

Existe uma enorme variedade de problemas de CMC com restrições adicionais e a terminologia associada a esse tipo de problemas varia muito consoante os autores. Uma parte significativa dos trabalhos sobre problema de CMC com restrições adicionais (CMCRA) tem-se baseado em técnicas de programação dinâmica. A programação dinâmica é uma técnica muito poderosa para resolver determinados tipos de problemas computacionais.
Nesta tese será considerado um tipo de problema de CMCRA: o problema de CMC com restrições de capacidade, que limitam o consumo de recursos ao longo do caminho.

O conteúdo desta seção é extraído de [18]. O problema do CMCRA surge como um subproblema na resolução do problema tratado nesta dissertação e em outros problemas como os de custos nos algoritmos *branch-and-price* para problemas de rotas de veículos com restrições adicionais. Em [18] existem três métodos para a sua resolução que enumeramos de seguida, mas apenas vamos apresentar detalhadamente um deles.
O primeiro método é o algoritmo de programação dinâmica exata com melhorias ao nível da procura bidireccional baseada em recursos limitados. O segundo método consiste num algoritmo de enumeração implícita onde os limites inferiores são calculados por programação dinâmica com relaxação estado-espaço. O terceiro método, chamado relaxação estado-espaço decremental, é uma combinação dos dois métodos anteriores: programação dinâmica exata e relaxação estado-espaço. Para maior detalhe ver [8, 6, 7].
A abordagem referida no primeiro método, que é a que vamos estudar, utiliza a programação dinâmica, através de um algoritmo de programação dinâmica exata que determina a solução ótima do CMCRA.

5.2.1 Definição do problema

O CMCRA é definido em [18] da seguinte forma. Considere um digrafo $G = (V, A)$, onde o conjunto de vértices V é constituído por um conjunto de vértices N representando $|N|$ clientes e dois vértices s e t que representam a origem e destino que serão neste caso um único ponto. Um custo não negativo c_{ij} é associado a cada arco $(i, j) \in A$. Os custos dos arcos correspondem a caminhos mais curtos e portanto, satisfazem a desigualdade triangular. Um prémio não negativo λ_i está associado a cada vértice $i \in N$, e um custo não negativo λ_0 está associado à origem e ao destino. Procuramos um caminho de s para t, visitando um subconjunto de outros vértices. Não são permitidos ciclos, o que define o caminho como elementar. O objetivo é minimizar o custo, dado pela diferença entre a soma dos custos dos arcos percorridos e a soma dos prémios recolhidos nos vértices visitados.

Esta definição do problema é comum a muitas versões do CMCRA decorrentes de diversos problemas de roteamento. Observe que se no problema considerado não existirem prémios associados aos vértices (como é o nosso caso), basta fazer $\lambda_i = 0$, para todo o $i \in V$. As restrições adicionais dependem, em geral, do tipo de aplicação a ser modelada. Um exemplo é especificado a seguir.

Capacidade

Uma variante do problema de CMC é quando há recursos limitados. Desta forma, as limitações de recursos são colocadas como restrições. Este é chamado de problema de CMC com Restrições de Capacidade (CMCRC). Geralmente, associado a cada arco de um grafo existe um peso d_{ij} que representa o consumo de um determinado recurso, que pode ser tempo, dinheiro, combustível, etc. A esse recurso é usualmente associada uma restrição que limita o seu consumo a uma

capacidade máxima, W. Um caminho p entre o vértice origem s e o vértice destino t diz-se admissível se verifica a restrição de capacidade

$$\sum_{(i,j)\in A} d_{ij}x_{ij} \leq W.$$

O CMCRC é NP-difícil, mesmo considerando o caso de grafos acíclicos com todos os pesos e custos positivos.

Este problema pode ser modelado como se segue.

$$\text{Minimizar} \sum_{(i,j)\in A} c_{ij}x_{ij}$$

Sujeito a

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ij} = \begin{cases} 1 & \text{para } i = s, \\ -1 & \text{para } i = t, \\ 0 & \text{para todos os outros vértices,} \end{cases}$$

$$\sum_{(i,j)\in A} d_i x_{ij} \leq W,$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A.$$

Modelo 2. Formulação do CMCRC como problema de PLI.

5.2.2 Programação dinâmica exata

Um dos métodos para a resolução do problema de CMCRC é o método de programação dinâmica exata. A programação dinâmica é uma técnica de resolução de problemas que divide o problema original em sub-problemas de resolução mais fácil. O problema de CMC com restrições de capacidade pode ser resolvido, construindo um caminho ótimo de s para t a partir de sub-caminhos ótimos. O paradigma de programação dinâmica segue os seguintes passos: remover um elemento do problema; resolver o problema menor e usar a solução do problema menor para adicionar o elemento removido de maneira adequada, produzindo uma solução para o problema maior. Vamos portanto descrever um algoritmo de programação dinâmica que resolve o problema de CMCRC. Este algoritmo é descrito em [18].

O problema de CMC definido num digrafo $G = (V, A)$ onde $V = \{1, ..., n\}$ pode ser resolvido com recurso à programação dinâmica, uma vez que um caminho ótimo do vértice 1 para o vértice n é construído a partir de sub-caminhos ótimos. Seja ainda s a origem e t o destino. O algoritmo que vamos abordar atribui estados a cada vértice: cada estado associado ao vértice i representa um caminho de s para i. Cada um desses estados inclui um vetor de consumo de recursos R cujas componentes representam as quantidades de cada recurso r usadas ao longo do caminho correspondente. Existe também um custo C associado a cada estado. A solução ótima corresponde ao estado de custo mínimo associado ao vértice t. O algoritmo estende repetidamente cada estado para gerar novos estados, até que todos os estados tenham sido estendidos em todos os caminhos admissíveis. A extensão do estado corresponde a juntar um arco adicional (i, j) ao caminho de s a i, obtendo um caminho de s para j. Com o intuito de garantir que o caminho não contém ciclos, isto é, de forçar a ser um caminho elementar, adiciona-se ao estado um recurso binário adicional para cada vértice $i \in V$. Existe apenas uma unidade disponível para cada recurso fictício que é consumida quando o vértice correspondente é visitado. O consumo dos recursos fictícios é indicado por um vetor S inicializado a 0 em todas as posições do vetor. Este vetor não dá informação sobre a ordem pela qual os vértices são visitados. Portanto, no algoritmo de programação dinâmica exata para o CMCRA, cada estado é representado por uma

etiqueta da forma (S, R, C, i) associada ao vértice i. Quando esta é estendida para gerar outra etiqueta admissível (S', R', C', j) associada ao vértice j, os vetores de consumo de recursos e de custo são atualizados e é verificada a admissibilidade do novo estado como se segue.

Custo

O custo é inicializado a 0 no estado inicial associado ao vértice s e é atualizado de acordo com a fórmula:

$$C' = C - \lambda_i/2 + c_{ij} - \lambda_j/2,$$

onde $\lambda_i = \frac{-\lambda_0}{2}$ se $i = s$ ou $i = t$.

Vejamos um exemplo desta equação de atualização de custo.

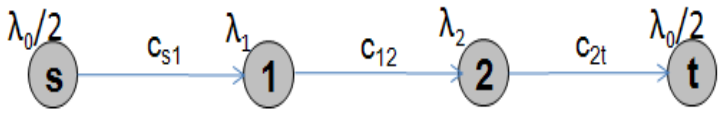

Figura 5.3: Equação de atualização de custo.

$$C' = 0 - \lambda_0/2 + c_{s1} - \lambda_1/2,$$

$$C'' = C' - \lambda_1/2 + c_{12} - \lambda_2/2,$$

$$C''' = C'' - \lambda_2/2 + c_{2t} - \lambda_0/2.$$

Então:

$$C''' = 0 - \lambda_0/2 + c_{s1} - \lambda_1/2 - \lambda_1/2 + c_{12} - \lambda_2/2 - \lambda_2/2 + c_{2t} - \lambda_0/2,$$

$$C''' = 0 - \lambda_0 - \lambda_1 - \lambda_2 + c_{s1}/2 + c_{12} + c_{2t}.$$

Recursos Fictícios

O vetor de recursos fictícios S é inicializado a 0 no vértice s e a regra de atualização é:

$$S_{k'} = \begin{cases} S_k + 1, & \text{se } k = j, \\ S_k, & \text{se } k \neq j. \end{cases}$$

O vetor de recursos fictícios é usado para garantir que o caminho é elementar. O estado (S, R, C, i) corresponde a um caminho elementar apenas se $S_k \leq 1$ para $k \in N$. Vejamos um exemplo desta equação de atualização de recursos fictícios:

De acordo com os diferentes tipos de recursos considerados, as regras de extensão e os testes de admissibilidade de R assumem diferentes formas. Veremos a seguir uma possível forma para R, que é usada na solução do CMCRC.

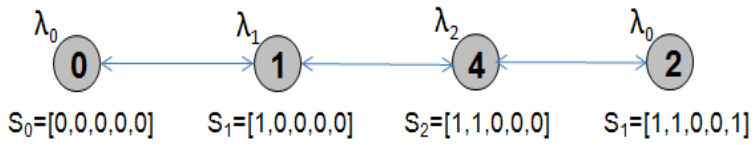

Figura 5.4: Equação de atualização de recursos fictícios.

Capacidade

A restrição de capacidade é modelada por um único recurso, que representa a quantidade de capacidade ainda disponível ao longo do caminho. Seja q a quantidade de recursos consumidos. Quando saimos do vértice s, todos os recursos estão disponíveis, ou seja, $q = 0$ no vértice s. Cada vez que um vértice j é visitado, q é aumentada de acordo com a procura desse vértice. Daí a regra de extensão ser $q' = q + d_j$. O estado (S, q, C, j) só é admissível se $q \leq Q$.
Esta técnica também pode ser generalizada para problemas com mais restrições. No entanto, o problema de CMCRA gera um elevado número de estados e a eficácia do algoritmo de programação dinâmica depende muito do número de estados gerados. Por isso, é essencial analisar os estados admissíveis, que podem conduzir à solução ótima, isto é, deve-se apenas gerar estados não dominados. Para este efeito, testes de domínio adequados são sempre realizados quando os estados são estendidos, de modo que o algoritmo registe apenas os estados não dominados. Um estado associado a um vértice $i \in V$, é caracterizado por (S, R, C, i), onde S é um recurso fictício, R uma capacidade do vértice origem s para o vértice i e C o custo do caminho. A definição de dominância é a seguinte: Sejam (S', R', C', i) e (S'', R'', C'', i) as etiquetas de dois estados associados ao vértice i, então (S', R', C', i) domina (S'', R'', C'', i) se:
$S' \leq S''$
$R' \leq R''$
$C' \leq C''$
e quando pelo menos uma das desigualdades é estrita. Por exemplo, o estado (0110,5,10,2) domina (1110,6,15,2) mas não domina o estado (1110,6,9,2).
De modo a gerar apenas estados não dominados é usual recorrer a técnicas de fixação de etiquetas em que para cada vértice só são geradas as etiquetas associadas a este tipo de estados. Note-se que o número de estados não dominados cresce rapidamente. Para contornar esta situação, certos autores propõem uma pesquisa bidireccional (do vértice origem para o vértice destino e vice versa). Neste caso, a meio da pesquisa é necessário combinar as listas de etiquetas geradas pelas duas pesquisas [18]. Outra forma de reduzir o número de etiquetas consiste em usar limites superiores e inferiores que permitem excluir estados a partir dos quais não é possível obter a solução ótima.

A ordem pela qual os estados são estendidos pode ser muito importante para a eficácia do algoritmo global. O algoritmo de programação dinâmica explicado, pode ser classificado como algoritmo *label-correcting*, uma vez que percorre uma lista de vértices e vê se há algum estado que possa ser estendido. Os vértices nunca são fechados, pois estamos sempre a tentar estender um estado para chegar até eles, tal como acontece nos algoritmos *label-correcting*. Os estados são explorados de acordo com os vértices aos quais estão associados. Todos os vértices são ciclicamente visitados e para cada vértice, do algoritmo estendem-se todos os estados que ainda não foram estendidos. Estados associados ao mesmo vértice podem ser classificados de acordo com um critério secundário, por exemplo, de acordo com o custo ou o consumo de um determinado recurso.

O problema de CMCRC pode então ser resolvido pelo Algoritmo 4 de programação dinâmica.

Algoritmo 3 Algoritmo de Programação dinâmica exata para a solução do CMCRC.

```
//Inicialização
Γ_s ← {(0, 0, 0, s)}
for all i ∈ V\{s} do
  (Γ_i ← ∅)
end for
E ← {s}
//Procura
repeat
  //Seleção de vértice
  Select i ∈ E
  //Extensão para a frente
  for all l_k = (S^k, R^k, C^k, i) ∈ Γ_i do
    for all j ∈ Δ_i^+ tal que S_j^i = 0 do
      q_j = q_i + d_j
      if q_j ≤ Q then
        S_i^j = 1
        C_j = C_i + c_ij − λ_j
        l_j = (S_i^j, q_j, C_j, j)
      else
        continua na próxima posição do ciclo for (continue)
      end if
      if l_j satisfaz as regras de domínio then
        Γ_j ← adicionar l_j a Γ_j
      end if
      if Γ_j foi alterado then
        E ← E ∪ {j}
      end if
    end for
    E ← E \ {i}
  end for
until E = ∅
```

Exemplo

Aplicamos de seguida um exemplo de modo a ilustrar a implementação do Algoritmo 4 de programação dinâmica, ao exemplo da rede dada na Figura 5.7.

Seja $G = (V, A)$ com c_{ij} associado ao arco (i, j), em que c_{ij} representa o custo para para atravessar o arco do vértice i para o vértice j.

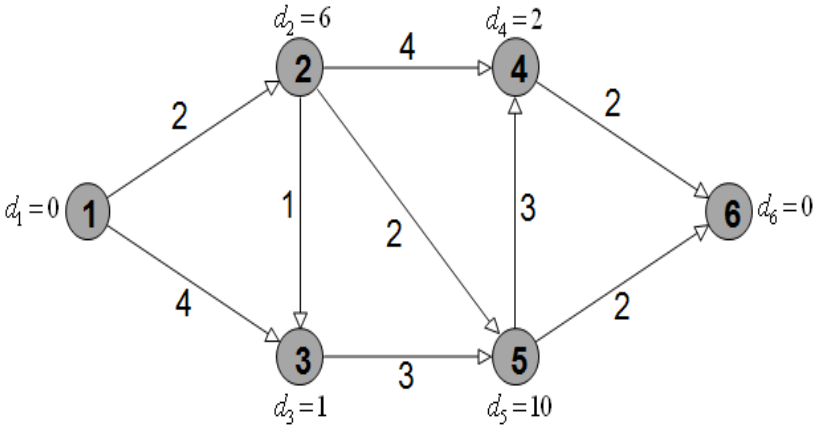

Figura 5.5: Grafo para a aplicação do Algoritmo exato para o CMCRC.

Decidimos usar o mesmo grafo usado para a resolução do Algoritmo de *Dijkstra* e acrescentámos-lhe a restrição de capacidade, de forma a comparar o comportamento dos algoritmos em situações em que o problema se torna um pouco mais complexo. Está também definida por d_i a procura em cada vértice i. Para todos os vértices i o tempo de serviço θ_i é igual a zero e não existem prémios nos vértices, ou seja, $\lambda_i = 0$. Além disso, vamos considerar a capacidade total $Q = 15$. Queremos calcular o caminho entre $s = 1$ e $t = 6$, sujeito à restrição de capacidade.

Consideremos então por observação do gráfico que:

$$d_1 = 0, \quad d_2 = 6, \quad d_3 = 1, \quad d_4 = 2, \quad d_5 = 10, \quad d_6 = 0.$$

Começamos por fazer a inicialização. Se a origem é 1, então:
$s = 1$
$\Gamma_1 = \{(000000, 0, 0, 1)\}$
$\Gamma_2 = \Gamma_3 = \Gamma_4 = \Gamma_5 = \Gamma_6 = \emptyset$
$E = \{1\}$

De seguida, selecionamos um vértice de E. Como para já só temos um único elemento, é esse que vamos analisar.

Seja $i = 1$.

Vamos fazer a extensão para a frente do único elemento que está em Γ_1.
$q^1 = q^1 + d_1 = 0 \leq 15$
$l_1 = (S^1, R^1, C^1, 1) = (000000, 0, 0, 1) \rightarrow l_1 = (S^1, q^1, C^1, 1) = (000000, 0, 0, 1)$
O nosso vetor $R^1 = q^1$ (capacidade) é inicializado a 0.

Os elementos que têm como predecessor 1, são o 2 e o 3, por isso, $\Delta_1^+ = \{2, 3\}$.
Vamos para cada um deles calcular a sua etiqueta.
$j = 2 \in \Delta_1^+$ e $S_2^1 = 0$

$q_2 = q_1 + d_2 = 0 + 6 = 6$ e $S_2^1 = 1$
$C_2 = 0 + 2 - 0 = 2$
$l_2 = (100000, 6, 2, 2)$ satisfaz as regras de domínio,
logo, $l_2 = (100000, 6, 2, 2)$ é adicionado a Γ_2.
$\Gamma_2 = \{(100000, 6, 2, 2)\}$
$E = \{1, 2\}$

De igual forma procedemos para $j = 3$.

$j = 3 \in \Delta_1^+$ e $S_3^1 = 0$
$q_3 = 1$ e $S_3^1 = 1$
$C_3 = 4$
$l_3 = (100000, 1, 4, 3)$ satisfaz as regras de domínio,
logo, $l_3 = (100000, 1, 4, 3)$ é adicionado a Γ_3.
$\Gamma_3 = \{(100000, 1, 4, 3)\}$
$E = \{1, 2, 3\} \rightarrow E = \{2, 3\}$

Neste momento temos os seguintes conjuntos não vazios:
$\Gamma_1 = \{(000000, 0, 0, 1)\}$
$\Gamma_2 = \{(100000, 6, 2, 2)\}$
$\Gamma_3 = \{(100000, 1, 4, 3)\}$

Como temos dois elementos de E, podemos selecionar qualquer um. Vamos selecionar o primeiro que foi introduzido neste conjunto.
Seja $i = 2$.

$l_2 = (S^2, q^2, C^2, 2) = (100000, 6, 2, 2)$
$q_2 \leq Q \rightarrow 6 \leq 15$
$\Delta_2^+ = \{3, 4, 5\}$
$j = 4 \in \Delta_2^+$ e $S_4^2 = 0$
$q_4 = q_2 + d_4 = 6 + 2 = 8$ e $S_4^2 = 1$
$C_4 = 2 + 4 - 0 = 6$
$l_4 = (110000, 8, 6, 4)$ satisfaz as regras de domínio,
logo, $l_4 = (110000, 8, 6, 4)$ é adicionado a Γ_4.
$\Gamma_4 = \{(1, 8, 6, 4)\}$
$E = \{2, 3, 4\}$
$j = 5 \in \Delta_2^+$ e $S_5^2 = 0$
$q_5 = q_2 + d_5 = 6 + 10 = 16 > 15$ Não satisfaz $q \leq Q$.
$E = \{2, 3, 4\}$
$j = 3 \in \Delta_2^+$ e $S_3^2 = 0$
$q_3 = 6 + 1 = 7$ e $S_3^2 = 1$
$C_3 = 2 + 1 - 0 = 3$
$l_3 = (110000, 7, 3, 3)$ satisfaz as regras de domínio,
logo, l_3 é adicionado a $\Gamma_3 = \{(110000, 1, 4, 3), (100000, 7, 3, 3)\}$
$E = \{2, 3, 4, 5\} \rightarrow E = \{3, 4, 5\}$

Seja $i = 3$.

$l_3 = (100000, 1, 4, 3)$
j=5 $\in \Delta_3^+ = \{5\}$ e $S_5^3 = 0$
$q_5 = q_3 + d_5 = 1 + 10 = 11 \leq 15$
$S_5^3 = 1$ e $C_5 = 4 + 3 - 0 = 7$
$l_5 = (101000, 11, 7, 5)$ satisfaz as regras de domínio,
logo, l_5 é adicionado a $\Gamma_5 = \{(101000, 11, 7, 5)\}$

$l_3 = (110000, 7, 3, 3)$
j=5 $\in \Delta_3^+ = \{5\}$ e $S_5^3 = 0$
$q_5 = q_3 + d_5 = 7 + 10 = 17 > 15$ Não satisfaz $q \leq Q$
$E = \{3, 4, 5\} \rightarrow E = \{4, 5\}$

Neste momento temos os seguintes conjuntos não vazios:
$\Gamma_1 = \{(000000, 0, 0, 1)\}$
$\Gamma_2 = \{(100000, 6, 2, 2)\}$
$\Gamma_3 = \{(100000, 1, 4, 3), (110000, 7, 3, 3)\}$
$\Gamma_4 = \{(110000, 8, 6, 4)\}$
$\Gamma_5 = \{(101000, 11, 7, 5)\}$

Seja $i = 4$.

$l_4 = (110000, 8, 6, 4)$
$j = 6 \in \Delta_4^+ = \{6\}$ e $S_6^4 = 0$
$q_6 = 8 + 0 = 8 \leq 15$
$C_6 = 6 + 2 = 8$
$l_6 = (110100, 8, 8, 6)$ satisfaz as regras de domínio,
logo l_6 é adicionado a $\Gamma_6 = \{(110100, 8, 8, 6)\}$
$E = \{4, 5, 6\} \rightarrow E = \{5, 6\}$

Seja $i = 5$.

$l_5 = (101000, 11, 7, 5)$
$j = 4 \in \Delta_5^+ = \{4, 6\}$ e $S_6^5 = 0$
$q_4 = 11 + 2 = 13$ e $C_4 = 7 + 3 = 10$
$l_4 = (101010, 13, 10, 4)$ é dominado por $(110000, 8, 6, 4)$, por isso não entra.
$j = 6 \in \Delta_5^+ = \{4, 6\}$ e $S_6^5 = 0$
Neste caso vamos usar o Γ_5 que não é ótimo (pois tem um pior custo
se partirmos do vértice 3 em vez do vértice 2 para 5) mas é a única solução admissível.
$q_6 = 11 + 0 = 11$
$C_6 = 7 + 2 = 9$
$l_6 = (101010, 11, 9, 6)$ satisfaz as regras de domínio,
logo l_6 é adicionado a $\Gamma_6 = \{(110100, 8, 8, 6), (101010, 11, 9, 6)\}$
$E = \{6\}$

Como 6 já não tem vértices sucessores, $E = \emptyset$ e terminamos por isso o algoritmo.
Encontramos assim a solução: $1 > 2 > 4 > 6$ com custo 8. A capacidade é de 8.

Capítulo 6

Algoritmo para o FCNDP-SPR

Considerando que em problemas como o FCNDP-SPR, fixar apenas uma aresta de cada vez causa pouco efeito na solução do problema, como foi verificado nas experiências em [11], o algoritmo do capítulo 4 pode ser eficiente para o FCNDP-SPR, uma vez que este fixa várias arestas, mergulhando na árvore de procura. Por isso, nesta dissertação, será adaptado o algoritmo do capítulo 4 para o nosso problema. Como visto no capítulo 4, precisamos resolver repetidas vezes dois problemas: o problema paramétrico definido pelas equações (4.11) a (4.15) e o problema do seguidor. Neste capítulo vamos mostrar como estes dois problemas estão definidos ao aplicarmos o algoritmo de enumeração implícita para resolver o problema FCNDP-SPR.

Relembrando a definição do problema que estamos a resolver, vimos que se $G = (V, E)$ é um grafo não orientado, onde V é o conjunto de vértices desse grafo e E o conjunto de arestas admissíveis que os conectam. Para cada aresta $e \in E$ associamos um custo fixo de abertura da aresta f_e e a cada arco $a \in A^E$ associamos um comprimento positivo c_a e um custo operacional da variável para o transporte de um dado produto através dela g_a^k (isto é, o custo variável de transportar o produto k pelo arco a), para cada produto $k \in K$. Então, a modelação do FCNDP-SPR como um problema 0-1 BLPP apresentado em [17], é definida como:

$$\min_{y} \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A^E} g_a^k x_a^k \tag{6.1}$$

$$y_e \in \{0, 1\}, \qquad \forall e \in E, \tag{6.2}$$

onde x é solução ótima para o problema

$$\min_{x} \sum_{k \in K} \sum_{a \in A^E} c_a x_a^k \tag{6.3}$$

Sujeito a

$$\sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b_i^k \qquad \forall i \in V, \forall k \in K, \tag{6.4}$$

$$x_a^k + x_{\bar{a}}^k \leq y_a \qquad \forall a \in A^E, \forall k \in K, \tag{6.5}$$

$$x_a^k \geq 0 \qquad \forall a \in A^E, \forall k \in K. \tag{6.6}$$

O problema de desenho da rede encontra uma rede para o transporte de $|K|$ mercadorias entre as suas respetivas origens e destinos. Cada mercadoria corresponde a um par OD. Seja (o_k, d_k) o par OD da mercadoria k, com $k \in K$.
Para cada produto $k \in K$ será definido um vetor de procura $b^k \in \{-1, 0, 1\}^{|V|}$ especificado como se segue:

$$b_i^k = \begin{cases} -1 & \text{se } i = d^k, \\ 1 & \text{se } i = o^k, \\ 0 & \text{caso contrário.} \end{cases}$$

Para cada arco $(i,j) \in A^E$ e para cada k com $k \in K$, definimos a variável de decisão binária:

$$x_a^k = \begin{cases} 1, & \text{se na solução a mercadoria perigosa } k \text{ passa pelo arco } a, \\ 0, & \text{caso contrário.} \end{cases}$$

Para cada aresta $e \in E$ definimos a outra variável de decisão binária:

$$y_e = \begin{cases} 1, & \text{se a aresta } e \text{ está acessível,} \\ 0, & \text{caso contrário.} \end{cases}$$

A região de restrição do problema formulado anteriormente é definida pelas restrições dos problemas interior e exterior, que é o conjunto de caminhos admissíveis para todas as mercadorias no grafo G. O conjunto admissível dos seguidores para cada decisão do líder é dado pelo conjunto de caminhos admissíveis num grafo restrito definido por $A(y) \subseteq A$, que é o conjunto de arcos de A tal que $y_e = 1$. O conjunto de reação racional é o conjunto de caminhos de custo mínimo do seguidor para cada decisão do líder. O conjunto sobre o qual o líder otimiza é a região induzível, e é o conjunto de todos os caminhos admissíveis para o líder e os seguidores, de modo que os caminhos de custo mínimo são selecionados pelos seguidores, dada a decisão do líder.

O modelo do problema paramétrico apresentado no capítulo 4 para o FCNDP-SPR é o seguinte:

$$\min \sum_{k \in K} \sum_{a \in A} c_a^k x_a^k \tag{6.7}$$

Sujeito a

$$\sum_{a \in \delta^+(i)} x_a^k - \sum_{a \in \delta^-(i)} x_a^k = b_i^k, \qquad \forall k \in K, \quad \forall i \in V, \tag{6.8}$$

$$x_a^k \leq y_e, \qquad \forall e \in E, \quad \forall k \in K, \quad \forall a \in A^E \tag{6.9}$$

$$x_a^k \in \{0,1\}, \qquad \forall a \in A^E, \quad \forall k \in K, \tag{6.10}$$

$$F(y_e, x_a^k) = \sum_{e \in E} f_e y_e + \sum_{k \in K} \sum_{a \in A} d^k g_a^k x_a^k \leq \alpha, \quad \forall a \in A^E, \quad \forall e \in E, \quad \forall k \in K, \tag{6.11}$$

$$\sum_{e \in E} y_e \geq \beta, \qquad \forall e \in E, \qquad y_e \in \{0,1\}, \quad \forall e \in E. \tag{6.12}$$

O problema paramétrico consiste em adicionar duas restrições adicionais ao problema de caminho mínimo, uma que obriga a existir um mínimo de β variáveis controladas pelo líder abertas (6.12) e uma outra garantindo que a função objetivo do líder, que traduz o risco total da rede, fica sempre limitada por um valor α (6.11). As restrições (6.8) asseguram o fluxo da mercadoria k desde a sua origem até ao seu destino. As restrições (6.9) asseguram que apenas os arcos selecionados pelo líder podem ser usados pelos seguidores.

Assim, o algoritmo de enumeração implícita vai resolvendo $|K|$ caminhos mais curtos com duas restrições adicionais, melhorando a cada caminho, sempre que possível o α e criando um artifício para evitar que no passo seguinte tenhamos a mesma solução, obrigando o seguidor a usar pelo menos β variáveis do líder. O problema paramétrico força o líder e o seguidor a entrarem em acordo para abrir pelo menos β arestas.

6.1 Algoritmo para o FCNDP-SPR

De seguida apresentamos o algoritmo de enumeração implícita para o nosso caso, onde chamaremos a atenção para as diferenças em relação ao apresentado no capítulo 4.
O algoritmo modificado está descrito no Algoritmo 5, onde as alterações e detalhes específicos à resolução do FCNDP-SPR em relação à versão original são acentuadas a vermelho.

Algoritmo 4 Algoritmo árvore de procura

Inicialização: cria raiz da árvore de procura
($k=0$; $S_0^+=\emptyset$; $S_0^-=\emptyset$; $S_0^0=\{1,\ldots,n_1\}$; $\alpha=+\infty$; $\beta=0$; $\underline{F}=+\infty$)
while $k\geq 0$ **do**
  $y_j=1$ para $j\in S_k^+$; $y_j=0$ para $j\in S_k^-$;
  Para os valores atuais de α e β encontrar solução admissível para [(4.11)-(4.15)]
  if encontrou solução admissível para [(4.11)-(4.15)] **then**
    Nomear a solução obtida como (y^k,x^k)
    //*bounding* (limites)
    Fixar y em y^k e resolver problema seguidor para encontrar $(y^k,\hat{x}^k)\in IR$ que torna mínimo o custo da função objetivo do seguidor - $F(y^k,\hat{x}^k)$
    Calcular $\underline{F}=min\{\underline{F},F(y^k,\hat{x}^k)\}$
    //$J=$ variáveis que não estavam fixas e fixámos em 1
    $J=\{j\in S_k^0 \text{ e } y_j^k=1\}$
    if $J\neq\emptyset$ **then**
      //*branching* (ramificação)
      backtracking=0;
      Incrementar k de cada vez que criar um nó
      Criar $|J|$ novos nós: juntar os elementos $j\in J$ a P_{k-1} na ordem ascendente, 1 a 1, para obter os novos nós disponíveis e o caminho final P_k
      $S_k^+=S_{k-1}^+\bigcup J$; $S_k^0=S_{k-1}^0\setminus J$; $S_k^-=S_{k-1}^-$;
      $\alpha=\underline{F}-1; \beta=1+|S_k^+|$;
    else
      $S_k^+=S_{k-1}^+$; $S_k^-=S_{k-1}^-$; $S_k^0=S_{k-1}^0$; $P_k=P_{k-1}$;
      $\alpha=\underline{F}-1$; $\beta=1+|S_k^+|$;
      Voltar ao início do while para resolver o mesmo problema com novos parâmetros.
    end if
  else
    Descartar nó atual e realizar *backtracking*
    backtracking=1;
  end if

  // *backtracking*
  if (backtracking==1) **then**
    while $k\geq 0$ **do**
      Voltar atrás desde o nó atual 1 nível (nó mais recentemente criado)
      if $(y_{j'}^k=1)$ **then**
        //Ramificar o seu complementar (Chamar a variável correspondente j')
        $k=k+1$;
        $y_{j'}^k=0$;
        Atualizar S_k^+,S_k^-,S_k^0,P_k, de acordo com as suas definições
        $\beta=0$;
        continue;
      end if
    end while
  end if
  if não existem mais nós **then**
    if F=valor muito grande **then**
      não existe solução admissível para [(2.4)-(2.7)]
    else
      declarar ponto admissível associado a solução ótima F
    end if
    break;
  end if
end while

Temos agora duas questões que precisam de ser resolvidas e serão apresentadas nas duas seções seguintes.

6.2 Solução para o problema paramétrico [(4.11)-(4.15)]

Nesta seção será apresentado como será resolvido no nosso caso o problema paramétrico.
Este é bem mais difícil de resolver do que o problema do seguidor. A dificuldade deste problema está relacionada com o fato de termos mais do que um caminho mínimo para calcular. Será apresentado um método de solução correto para resolver este problema. Temos um problema com variáveis do líder e do seguidor (y e x respetivamente) misturadas nas inequações, o que impede que quebremos o problema em sub-problemas mais fáceis de resolver. Resolver este problema de forma exata, seria muito custoso. Por isso, com base na Proposição 2 do capítulo 4, que afirma que resolver o problema [(4.11)-(4.15)] de forma heurística pode ser suficiente, pensámos em construir uma heurística.

Propomos usar uma heurística simples para encontrar uma solução admissível, utilizando um algoritmo que é conhecidamente eficiente para o problema de caminho mínimo com restrições adicionais, que vai resolvendo $|K|$ caminhos mais curtos com duas restrições adicionais, melhorando a cada caminho, sempre que possível o α e forçando o líder e o seguidor a entrarem em acordo para abrir pelo menos β arestas. Esta é só uma sugestão simples de resolver o problema paramétrico [(4.11)-(4.15)], utilizando um algoritmo que é conhecidamente eficiente para o problema de CMCRC apresentado no capítulo 5.

Se a heurística responder que o problema paramétrico é inadmissível, então, precisaríamos resolver de forma exata o problema paramétrico, de forma a encontrar a otimalidade. Se isso for feito, o nosso algoritmo de enumeração dará uma solução exata para o problema FCNDP-SPR. Se isso não for feito, no final, teremos apenas uma solução heurística. Usando a heurística pretendemos evitar ter que resolver o problema paramétrico de forma ótima.

Apresentamos de seguida um exemplo da heurística.

Suponhamos que temos como capacidade máxima $\alpha = 20$ e a seguinte rede com custos nos arcos (f_e, g_a^k, c_a^k):

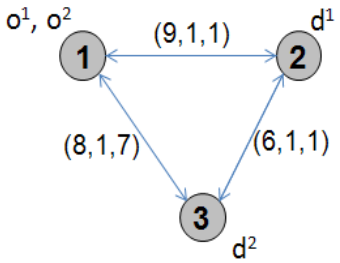

Figura 6.1: Grafo para a aplicação da heurística.

Para a mercadoria 1:
$k = 1$
$\alpha' = \alpha = 20$
$fy = 9$ e $x^1 g^1 = 1$

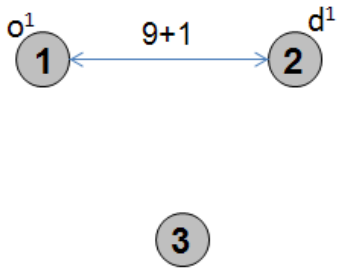

Figura 6.2: Aplicação da heurística para a mercadoria 1.

$fy + x^1 g^1 = 10 \leq 20$

Para a mercadoria 2:
$k = 2$
$\alpha'' = \alpha - \alpha' = 20 - 10 = 10$

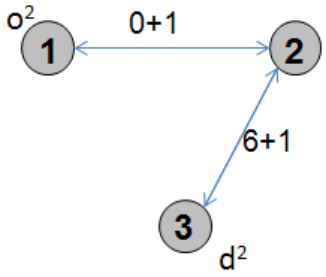

Figura 6.3: Aplicação da heurística para a mercadoria 2.

Neste caso, os custos fixos da mercadoria $k = 1$ já não vão ser considerados, uma vez que já estão a contar na solução anterior.
$fy = 0 + 6 = 6$ e $x^2 g^2 = 1 + 1 = 2$
$fy + x^2 g^2 = 8 \leq 10$

O custo final é portanto 18 e o líder tem de abrir as arestas $(1, 2)$ e $(2, 3)$ com custo fixo de as abrir igual a 15.
A mercadoria 1 será transportada pela aresta $(1, 2)$ e a mercadoria 2 será transportada pelas arestas $(1, 2), (2, 3)$.

6.3 Solução para o problema do seguidor

No algoritmo de enumeração implícita também é necessário resolver o problema do seguidor que consiste em minimizar a soma dos vários caminhos mínimos.
Resolver o problema do seguidor de encontrar $(y^k, \hat{x}^k) \in IR$ que torna mínimo o custo da função objetivo do seguidor é fácil. Vamos recorrer ao algoritmo de *Dijkstra*, apresentado no capítulo 5, calculando um caminho mínimo para cada mercadoria. A versão do *Dijkstra* usada, satisfaz a condição suficiente da Proposição 1 no capítulo 4. Ou seja, o seguidor escolhe sempre $x \in \Psi(y)$ que maximiza $F(y, x)$ para y fixo. Isto significa que nesta implementação não são quebrados os empates arbitrariamente. No caso de vários caminhos mínimos será escolhido o que é melhor

para o líder.

Capítulo 7

Conclusões

Como vimos, o FCNDP-SPR possui aplicações importantes como o transporte de materiais perigosos. Muitos dos materiais transportados, apesar de serem potencialmente nocivos para o ambiente e para a população, são essenciais para o desenvolvimento industrial. Devido à magnitude potencial de acidentes para a população e para o meio ambiente, o público é muito sensível aos perigos desta atividade. Daí, o risco associado a incidentes envolvendo embarques de materiais perigosos ter encontrado uma atenção considerável por parte do governo, incentivando a pesquisa sobre este tema. Uma forma dos responsáveis pela segurança populacional e ambiental mitigarem os riscos de transporte de materiais perigosos é limitá-los a um subconjunto de estradas disponíveis. Muitas vezes designam-se "rotas de mercadorias perigosas". Temos então o governo principalmente interessado em minimizar o risco total induzido pela execução de um determinado conjunto de pedidos de envio de materiais perigosos, enquanto as transportadoras estão sobretudo interessadas na minimização de custos. Esta situação dá origem a um problema de projeto de dois níveis, com o governo que representa o nível de decisão do líder e as transportadoras representando o nível de decisão dos seguidores.

Desde o estudo inicial de *Kara* e *Verter* [14], existiram vários estudos na literatura, que abordam diferentes métodos para resolver o 0-1 BLPP, como foi referido na introdução. Dos métodos encontrados na literatura para a resolução do FCNDP-SPR, alguns são métodos exatos e outros heurísticos propostos para resolver a aplicação do transporte de materiais perigosos [14, 2, 4, 10] e uma busca Tabu proposta em [11].

Encontrámos em [17] uma proposta de um algoritmo de enumeração implícita para resolver de forma exata problemas do tipo 0-1 BLPP. Nesta dissertação estudamos este algoritmo e adaptámo-lo para resolver o FCNDP-SPR. O algoritmo apresentado em [17] começa por criar a raiz da árvore de procura, tenta encontrar na iteração geral uma solução para um problema paramétrico e em caso positivo, fixa as variáveis do líder e resolve o problema do seguidor, de forma a tornar mínimo o custo da função objetivo do seguidor. Faz de seguida o *branching* das variáveis a introduzir na árvore e volta à iteração geral. Quando na iteração geral não encontra solução admissível, faz o *backtracking*. Este algoritmo termina quando já não existe mais nenhum nó da árvore a analisar. Para adaptar este algoritmo para a resolução do FCNDP-SPR precisamos resolver o problema paramétrico, e para isso propomos o uso de uma heurística que vai resolvendo os vários problemas de CMCRA definidos pelos parâmetros α e β que definem o problema paramétrico, atualizando sempre estes parâmetros. Caso esta não encontre uma solução admissível, devemos então resolver o problema utilizando um método exato.
Para além disso, pretendemos também resolver o problema do seguidor, isto é, pretendemos minimizar a soma dos vários caminhos mínimos. Para isso, podemos usar o algoritmo de *Dijkstra*.

Esta dissertação apresenta apenas uma proposta para a resolução do FCNDP-SPR utilizando o algoritmo proposto em [17]. Um estudo mais aprofundado sobre a resolução do problema para-

métrico e da regra de *branching* usada, pode gerar uma versão bastante eficiente deste algoritmo.

Bibliografia

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993.

[2] E. Amaldi, M. Bruglieri, and B. Fortz. On the hazmat transport network design problem. *Lecture Notes in Computer Science*, 6701:327–338, 2011.

[3] M.S. Bazaraa, J.J. Jarvis, and H.D. Sherali. *Linear Programming and Network Flows*. Wiley, 2009.

[4] L. Bianco, M. Caramia, and S. Giordani. A bilevel flow model for hazmat transportation network design. *Transportation Research Part C: Emerging Technologies*, 17:175–196, 2009.

[5] J. W. Billheimer and P. Gray. Network design with fixed and variable cost elements. *Transportation Science*, 7:49–74, 1973.

[6] J. Bramel and Simchi-Levi. *Set-covering-based algorithms for the capacited VRP, The vehicle-routing problem*, chapter 8, pages 85–106. NetworK Routing, Handbooks in Operations Research and Management Science, 2002.

[7] J.F. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon, and F. Soumis. Vrp with time windows, the vehicle-routing problem. *SIAM Monographs on Discrete Mathematics and its Applications*, pages 157–193, 2002.

[8] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. *Time constrained routing and scheduling*, chapter 8, page 35–139. Network Routing, Handbooks in Operations Research and Management Science, 1995.

[9] E. Erkut and O. Alp. Designing a road network for dangerous goods shipments. *Computers and Operations Research*, 34:1241–1242, 2007.

[10] E. Erkut and F. Gzara. Solving the hazmat transport network design problem. *Computers and Operations Research*, 35:2234–2247, 2008.

[11] R.M.V. Figueiredo, M. Labbé, and A. Mauttone. A tabu search approach to solve a network design problem with user-optimal flows. In *In VI ALIO/EURO Conference on Combinatorial Optimization*, Buenos Aires, 2008.

[12] K. Holmberg and D. Yuan. Optimization of internet protocol network design and routing. *Networks*, 43:39–53, 2004.

[13] E. Iakovou, C. Douligeris, H. Li, C. Ip, and L. Yudhbir. A maritime global route planning model for hazardous materials transportation. *Transportation Science*, 33:34–48, 1999.

[14] B.Y. Kara and V. Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, 38:188–196, 2004.

[15] R.L. Keeney. Designing a road network for dangerous goods shipments. *Operations Research*, 28:527–534, 1980.

[16] G.F. List and P.B. Mirchandani. An integrated network/planar multiobjective model for routing and siting for hazardous materials and wastes. *Transportation Science*, 25:146–156, 1991.

[17] J.T. Moore and J.F. Bard. An algorithm for the discrete bilevel programming problem. *Operations Research*, 39:419–435, 1992.

[18] G. Righini and M. Salani. New dynamic programming algorithms for the resource constrained elementary shortest path problem. *Wiley InterScience, Inc.*, 51:155–170, 2007.

[19] S.N. Silva. *Problemas do caminho mais curto com restrições adicionais*. Master dissertation, Universidade de Aveiro, 2008.